UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

THESE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES 1
COMUE **U**NIVERSITE **B**RETAGNE **L**OIRE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par

Abd-El-Karim KECIR

Performance Evaluation of Urban Rail Traffic Management Techniques

Thèse présentée et soutenue à Rennes, le 05/07/2019
Unité de recherche : UMR6074 IRISA

Rapporteurs avant soutenance :

Béatrice BERARD	Professeur, HdR	— Université Pierre et Marie Curie
Stefan HAAR	Directeur de recherche, HdR	— INRIA Paris–Saclay

Composition du Jury :

Président :	Albert BENVENISTE	Directeur de recherche	— INRIA Rennes – Bretagne Atlantique
Examinateurs :	Anne BOUILLARD	Maître de conférences	— NOKIA Paris–Saclay
	Walter SCHÖN	Professeur	— Université de Technologie de Compiègne
Dir. de thèse :	Loïc HELOUËT	Chargé de recherche	— INRIA Rennes – Bretagne Atlantique
Co-dir. de thèse :	Pierre DERSIN	Directeur RAM&PHM	— ALSTOM TRANSPORT

This humble work is dedicated to my dear friend Rafik M. ADICHE.

“La semplicità è la sofisticazione finale.”
Simplicity is the ultimate sophistication.

— Leonardo di ser Piero DA VINCI[1]

[1] attributed to.

Remerciements

Je tiens ici, et en premier lieu, à exprimer mes plus profonds sentiments de gratitude à M. Loïc HÉLOUËT. Sa générosité d'esprit, sa patience et son sens de la pédagogie auront indélébilement marqué cette aventure professionnelle et scientifique. Merci, Loïc, de m'avoir inlassablement guidé et soutenu tout au long de ce parcours et vers son achèvement ; merci de l'avoir agrémenté de tant de moments de complicité, et merci d'en avoir fait une expérience humaine inouïe.

Je remercie tout aussi chaleureusement M. Pierre DERSIN qui a assuré, durant ce projet, toutes les ressources nécessaires à son bon déroulement. C'est également sa flexibilité et son expérience qui ont ouvert la voie à toutes les directions que ce projet a pu prendre ; je tiens ainsi à l'en remercier tout particulièrement.

Merci à M. Éric FABRE d'avoir bien voulu m'accueillir au sein de l'équipe SUMO, de m'avoir fait confiance avec cette mission du projet commun ALSTOM–INRIA, mais aussi pour son soutien pendant les périodes difficiles de ce parcours.

Mes remerciements vont également aux membres du jury de soutenance de cette thèse qui m'ont fait l'honneur d'accepter d'en faire part : Mme Béatrice BÉRARD et M. Stefan HAAR pour leurs rapports et la pertinence de leurs retours mais aussi pour leur soutien durant les derniers mois, et Mme Anne BOUILLARD, M. Albert BENVENISTE et M. Walter SCHÖN pour avoir bien voulu évaluer ce travail.

Mes sentiments d'amitié et, de même, de gratitude vont à l'ensemble des membres de l'équipe SUMO, Éric BADOUEL, Nicolas BERTHIER, Nathalie BERTRAND, Thierry JÉRON, Hervé MARCHAND, Nicolas MARKAY, Christophe MORVAN et Ocan SANKUR que j'ai eu le grand plaisir de côtoyer. Un grand merci va à Laurence DINH pour sa sympathie singulière.

Merci à MM. Bruno ADELINE et Denis MULARD pour leurs précieux commentaires et suggestions et, plus généralement, à tous mes collègues d'ALSTOM pour leur collaboration et compagnie.

Parallèlement à cette thèse, j'ai eu le privilège de suivre un programme doctoral complémentaire de l'Institut européen de l'innovation et de la technologie (EIT) qui promeut l'entrepreneuriat et le rapprochement entre les entreprises et le monde académique ; je profite ainsi de cet élan pour saluer et remercier les responsables du pôle rennais, MM. Yvonnick DAVID et Álvaro I. PIÑA-STRANGER et Mme Stéphanie GAUVIN, mais aussi mes condisciples Mamadou D. BALDÈ, Pauline BOLIGNANO, Panagiotis 'Panos' MAVRIDIS, Laudin A. MOLINA et Marcelino RODRÍGUEZ-CANCIO.

Que mes collègues, Mme Cécile BIDAN, M. Patrice GÉLIN, M. Michel HURFIN et M. Stéphane UBÉDA, du comité de pilotage du programme de sensibilisation à l'entrepreneuriat INRIA HORIZON STARTUP trouvent ici l'expression de mes sentiments les plus chaleureux.

En mai 2016, je me suis déplacé en Italie pour une collaboration avec le laboratoire AUTORI de l'université de Rome III. Je remercie chaleureusement les professeurs Andrea D'ARIANO et Dario PACCIARELLI, et aussi Marcella SAMÀ d'avoir bien voulu accepter de

me recevoir, de l'accueil chaleureux qu'ils m'ont réservé mais surtout de toutes les glaces italiennes que nous avons partagées pendant mon séjour.

C'est principalement mon intérêt pour l'étude des systèmes à événements discrets qui m'a amené à poursuivre ce projet doctoral ; tout le mérite revient à mes chers enseignants M. Abdelhay BENHALLA de l'université de Boumerdès et M. Michel COMBACAU, Mme Euriell LE CORRONC et Mme Pauline RIBOT de l'université de Toulouse III qui m'ont fait découvrir et apprécier ce domaine. Je tiens ici à leur en témoigner ma plus vive reconnaissance.

Un très grand merci à tous les aspirants rikishis ayant lutté à mes côtés pendant ces dernières années au sein de l'équipe SUMO ; notamment à Matthieu PICHENÉ, pour tous les problèmes d'échecs que l'on a pu résoudre ensemble, Hugo BAZILLE, pour sa sympathie et son amitié, Engel LEFAUCHEUX, pour sa bonne humeur contagieuse, Robert F. Jr. NSAIBIRNI et Sanaa MAIROUCH, pour leur légèreté d'esprit, et Dhananjay RAJU, pour son dynamisme transmissible.

L'un des moments les plus mémorables de ces dernières années est indubitablement celui où j'ai pu porter pour la deuxième fois Heman, l'adorable fils de mon cher ami Sucheendra K. 'Suchee' PALANIAPPAN, après un voyage de plus de 24 heures pour Tokyo, à l'autre bout du monde. Suchee, merci pour tous les moments que nous avons passés ensemble, à Rennes, à Saint-Malo, à Kawasaki et ailleurs.

Étant donné la nature du sujet de cette thèse, qualifier ce parcours de voyage serait une métaphore bien appropriée. Dans ce même esprit, je salue Ajit RAI qui a emprunté le même train en prenant part au projet commun ALSTOM–INRIA comme autre camarade doctorant.

Je salue pareillement Flavia PALMIERI, dont l'arrivée dans l'équipe a remarquablement ravivé l'ambiance, et par la même occasion tout le groupe de doctorants : Sihem CHERRARED, Erij ELMAJED, Rômulo MEIRA-GÓES, Adrián PUERTO AUBEL et Rituraj SINGH.

Comment pourrais-je oublier les moments passés en compagnie de Roberto CELLETTI et de Silvia DI NARDO à Rome, aussi brefs qu'ils étaient mais si agréables. Je leur transmets ici mes sentiments d'amitié.

Je remercie aussi mes amis de longue date Yacine M. BERRANEN, Salim DOUMAZ, Mohammed GROUNE, Nabil LASSOUANI, Sofiane OUDJEDI et Wassim YAKOUB pour leurs encouragements. Qu'ils soient persuadés que leur amitié est appréciée à sa juste valeur.

C'est avec un grand sentiment de nostalgie que je salue mes camarades de Toulouse, Tomas ANDRADE DIAZ, Marie AVIGNON, Harry CHAUDAT, Zoé LAFOREST, Bastien MIRAL et le brillant Ghyslain MAITRE.

Pour leur présence continue, virtuelle certes, et pourtant si... explosive, je transmets ici un salut fraternel à Amine NAILI et Zineddine SAHRAOUI.

Que ma famille, au sens large, trouve ici le témoignage de mes ineffables sentiments. Un merci tout particulier va à mon oncle Habib ainsi qu'à Saadia SAIDANI, Hamida M. SADLI et Ayoub M. NEGGAZ. Mes parents, ma sœur Naïma et mon frère Naïm, merci de votre soutien et amour inconditionnels.

Enfin, merci à toutes celles et tous ceux qui ont contribué, de près ou de loin, d'une manière ou d'une autre, à l'accomplissement de ce travail.

Je clos cette section en remerciant l'ensemble des institutions qui ont rendu possible la réalisation de ce travail, à savoir : l'université de Rennes I (UR1), l'université de Rome III (UR3), l'Institut de recherche en informatique et systèmes aléatoires (IRISA), l'Institut européen de l'innovation et de la technologie (EIT), l'Association nationale de la recherche et de la technologie (ANRT), INRIA et ALSTOM.

Résumé

Mots clés : *systèmes ferroviaires, régulation du trafic, réseaux de Petri, évaluation de performance, simulation stochastique, tables horaires, planification*

Problématique

De par leur efficacité et leur haute fréquence de service, les réseaux ferroviaires urbains constituent une solution de mobilité privilégiée dans les grandes villes. Ce haut niveau de demande requiert, par conséquent, le maintien d'une qualité de service adéquate. Les opérateurs de ces systèmes emploient différents outils et techniques pour une gestion efficace du trafic. Nous nous intéressons, dans ce travail, à l'analyse et l'évaluation de performance de ces méthodes.

Dans notre contexte, un système ferroviaire peut être vu comme une configuration de voies décrivant la topologie, un ensemble de trains et un planning de circulation de ces trains pendant la journée. Une journée d'exploitation est décrite par ce qu'on appelle une table horaire, c.-à-d. un planning dans lequel sont référencés toutes les dates désirées de départs et d'arrivées des trains des et aux stations. Or, ce planning est une vision idéalisée de la réalité et n'est jamais parfaitement respecté. En effet, pendant une journée d'exploitation, la circulation des trains est très souvent perturbée par différents événements, comme par exemple, des pannes matérielles, un malaise d'un passager ou encore plus fréquemment, des passagers qui retiennent les portes. Ces événements perturbateurs engendrent des arrêts indésirables des trains ; ce qui, non seulement, retarde les départs ou arrivées de ces trains, mais en plus, peut créer des conflits dans la table horaire entre les différents événements à venir. Lorsque le retard est local et de courte durée, il peut facilement être rattrapé en écourtant le temps de stationnement à la station suivante du train affecté, ou en choisissant un profil de vitesse plus véloce. Toutefois, il arrive que, dans certains cas, ce retard ne soit pas récupérable sur une seule station, et mène vers des dates de passage qui rentrent en conflit avec celles d'autres trains, notamment au niveau des jonctions.

Afin de pallier aux retards, les opérateurs des systèmes ferroviaires utilisent un ensemble de politiques de régulation du traffic, chaque politique visant à accomplir un objectif bien défini. Des exemples de politiques de régulation sont : rattrapage du retard (ou de l'avance) au plus tôt en jouant sur les temps de course et de stationnement, égalisation des headways[2] entre les trains, ou des politiques plus élaborées à objectifs multiples. Le but final de la régulation du trafic consiste à fournir un service qui satisfait au mieux un ensemble d'indicateurs de performance. Or, les politiques de régulation du trafic traditionnellement utilisées sont, pour la plupart, définies sous formes de règles, et construites et validées sur la base d'une approche empirique. Par conséquent, leur optimalité n'est pas prouvée et donc pas garantie. C'est dans ce cadre précis que viens ce travail pour fournir des outils de vérification et d'évaluation de performance des politiques de régulation.

Par ailleurs, la supervision du trafic ferroviaire se base sur des plans horaires utilisés comme références. Nous nous intéressons donc, additionnellement, à des questions de réalisabilité de ces plans horaires par les systèmes dans un contexte temporel et probabiliste.

[2] Temps entre deux passages successifs de trains par le même point.

Modèles pour la circulation des trains

Un réseau ferroviaire peut être vu comme un système à événements discrets (SED), c.-à-d. un système dans lequel l'espace d'états est discret et la transition d'état est déclenchée par un événement et non pas par le temps, contrairement aux systèmes à temps continu ou à temps discret (discrétisé). Considéré en tant que tel, un réseau ferroviaire peut alors être modélisé par un réseau de Petri, un modèle mathématique (et graphique) bien adapté à la représentation des SEDs. Cependant, le modèle standard des réseaux de Petri n'est pas suffisant pour faire apparaître le temps et la stochasticité d'une manière explicite. En effet, la considération de ces deux aspects-ci est indispensable pour traîter des problèmes principalement dus aux retards aléatoires dont le système est sujet.

Dans un système ferroviaire urbain, les trains circulent avec des vitesses pouvant atteindre les 80 km/h. A de telles vitesses, le faible coefficient d'adhérence des roues sur les rails pénalise les durées de freinage. Afin d'éviter toute collision entre les trains circulant dans la même direction, on s'assure de réserver à tout instant une portion de la voie exclusive à chaque train, pour qu'il n'y ait jamais deux trains (ou plus) dans une même portion. Cette technique de réservation des voies est appellé le cantonnement. Il existe deux méthodes de cantonnement : le cantonnement fixe, et le cantonnement mobile (déformable)

Cantonnement fixe. C'est une méthode classique qui consiste à découper les voies en portions et d'allouer une ou plusieurs portions successives à chaque train d'une manière exclusive. Comme abstraction d'un système ferroviaire à cantonnement fixe, nous utilisons le modèle des réseau de Petri temporels stochastiques [39], mais avec une sémantique élémentaire. Dans ce modèle, des jetons (marquage) représentent les trains, des places représenteront des états (p. ex. train en stationnement, train en course.), et des transitions représentent les événements de départ et d'arrivée. Un intervalle de temps est associé à chaque événement et représente les durées possibles devant s'écouler avant que cet événement ne puisse se produire (p. ex. temps de stationnement, temps de course.). De même, une fonction de répartition de probabilité est associée à chaque événement et est définie sur son intervalle. C'est cette fonction qui va permettre de générer des retards aléatoires sur les départs et arrivées. La sémantique élémentaire, quant à elle, permet d'interdire que deux jetons (trains) occupent la même place, à tout instant.

Cantonnement mobile. Le cantonnement fixe est une méthode robuste, mais elle peut être trop pessimiste par rapport aux capacités réelles du système en termes de densité des trains pouvant circuler en même temps. Le cantonnement mobile est une technique plus moderne qui permet de définir des enveloppes autour des trains à tout instant. Il assure une distance entre les trains en bannissant les contacts entre toute paire d'enveloppes. Nous proposons un modèle qui simplifie le cantonnement mobile : les réseaux de Petri à trajectoires. C'est un modèle de réseaux de Petri dans lequel les places contiennent des trajectoires au lieu de simples jetons. Le modèle permet d'avoir plusieurs trajectoires (et donc trains) dans la même place, mais impose qu'une distance soit toujours assurée entre chaque paire de trajectoires. Il autorise donc des headways inférieurs aux headways minimums dans un système modélisé par un réseau de Petri classique.

Simulation de la régulation du trafic

L'approche adoptée pour l'analyse de la régulation est la simulation. Cependant, un modèle de circulation des trains, tel que défini ici, ne permet pas, à lui seul, de simuler des actions de contrôle-commande. Pour ce faire, il faut définir un modèle de table horaire ainsi qu'un modèle pour la régulation. Nous nous sommes donc appuyés sur une formu-

lation à base de graphe de contraintes pour représenter la table horaire de référence. Les nœuds du graphe modélisent les événements (départs et arrivées) et portent leurs dates d'occurrence désirées ou effectives, et les arcs du graphe modélisent les contraintes de précédence entre les événements. Quant à la régulation, elle est encodée telle quelle, sous forme d'algorithme, et communique avec le modèle de circulation des trains (réseau de Petri) et la table horaire (graphe de contraintes) pour envoyer des consignes de régulation. Ce modèle modulaire réseau-graphe-commande permet d'effectuer des simulations aléatoires dans lequelles (a) le comportement désiré est dicté par le graphe, (b) le comportement réel est simulé par le réseau, et (c) le contrôle-commande par l'algorithme de régulation. Un outil de simulation a été développé à partir de ce modèle : SIMSTORS (pour simulateur de systèmes stochastiques sous régulation). Il a été testé sur une ligne d'un réseau ferroviaire urbain réel, et a permis de simuler, en une quarantaine de secondes, 4 heures de trafic, avec 50 trains circulant sur 24 stations ; ce qui constitue un gain de $\times 360$ en comparaison avec une simulation temps-réel.

Evaluation de performance

L'évaluation de performances d'une politique de régulation se fait à travers l'observation des résultats de simulation. Après la construction du réseau de Petri correspondant au système étudié, la description d'une table horaire de référence, et le choix d'une politique de régulation, une simulation du système sur toute la durée de la table horaire peut être lancée. Toutefois, dans un système stochastique, la considération d'un échantillon aléatoire, c.-à-d. une seule exécution aléatoire du système, ne permet pas de faire des observations concluantes. En effet, un échantillon ne peut pas résumer, à lui seul, le comportement global du système, et peut même, dans certains cas, fournir des résultats extrêmes, avec des probabilités d'occurrence faibles (donc rares). La bonne approche est la méthode de Monte-Carlo qui consiste à procéder à un grand nombre n d'exécutions aléatoires et de calculer ensuite une moyenne $\bar{x}$ d'une quantité observée en sortie de ces exécutions. Cependant, ceci n'est pas suffisant car cette moyenne est une moyenne empirique et dépend fortement du nombre d'exécutions n ; elle ne reflète pas forcément la moyenne théorique (réelle) μ. Selon la loi des grandes nombres, la moyenne empirique $\bar{x}$ tend vers la moyenne théorique μ quand n tend vers $+\infty$. Sur la base du théorème central limite [66], il est possible de calculer un intervalle I, avec une certitude $c \in [0,1]$ que la moyenne théorique soit dans cet intervalle. L'intervalle I est fonction du niveau de certitude, qui est choisi arbitrairement et préallablement au calcul de I. Afin d'obtenir des intervalles étroits avec un grand niveau de certitude, il est nécessaire d'effectuer un grand nombre de simulations n.

La qualité du service ferroviaire est jaugée par des indicateurs clés de performance (ICP). Ces ICPs peuvent avoir la forme d'une mesure de la ponctualité des trains (p. ex. le nombre de missions de trains s'étant effectuées avec un retard supérieur à x minutes), d'une mesure de la régularité du service (p. ex. le nombre de départs s'étant effecutés avec des headways proche des headways planifiés, avec une tolérance de x minutes), ou d'autres mesures qui quantifient le retard, l'économie d'énergie, le confort des passagers, etc. Afin d'évaluer l'efficacité d'une politique de régulation, il suffit de définir un ou des ICPs, de lancer des simulations avec cette politique de régulation, et d'observer le niveau de satisfaction de ces ICPs par le système régulé.

Réalisabilité des plans horaires

La construction d'une table horaire de référence optimale pour un système ferroviaire urbain est un problème complexe. En effet, quand bien même certains paramètres tels que les limites de vitesses des trains et les contraintes dues à la topologie du réseau sont bien connus et maitrisés, ce n'est pas le cas pour certains facteurs externes tels que la distribution des passagers pendant une journée sur le réseau ou leur comportement, les pannes matérielles, etc. En considérant les vitesses des trains, leur capacités de freinage, leurs longeurs, leur itinéraire désirés ainsi que d'autres paramètres, un système peut être modélisé par un ensemble de contraintes. La construction d'une table horaire équivaudrait ensuite à résoudre ce système de contraintes. Cependant, les modèles obtenus sont souvent très importants en taille. Ceci oblige les experts en construction de table horaire à faire des simplifications et des approximations. Il en résulte que la solution obtenue ne soit pas optimale. Quand un système et un plan horaire correspondant sont conçues séparement, rien ne garantit que ce système pourra bien réaliser le plan horaire considéré. En raison de la stochasticité inhérentes aux systèmes de transport, qui peuvent être vus comme des systèmes stochastiques à variables aléatoires continues, la probabilité de réaliser parfaitement un plan horaire se trouve nulle. Il est donc plus pertinent de se poser la question : *« Un système peut-il réaliser un plan horaire avec un certain niveau de tolérence aux erreurs prédéfini ?»*, ou encore mieux : *« Un système peut-il réaliser un plan horaire avec un certain niveau de tolérence aux erreurs prédéfini et avec une probabilité significative ?»* Afin de répondre à ces questions, nous proposons, dans ce travail, une approche analytique basée sur le dépliage de réseaux de Petri temporels stochastiques à sémantique élémentaire (RdPSTe).

Tout d'abord, nous modélisons le système étudié par un RdPSTe, et le plan horaire qu'il doit satisfaire par un graphe. Les nœuds du graphes sont datés et représentent les dates désirées d'ocurrence des événements. Nous déplions ensuite le RdPSTe structurellement, c.-à-d. sans tenir compte du temps et de la stochasticité, à la manière d'ESPARZA et al. [26]. Ensuite, dans un second temps, nous associons un ensemble de contraintes aux nœuds du dépliage (conditions et événements). Ces contraintes assurent la précédence, l'urgence, la cohérence du comportement temporel avec le réseau, et interdisent l'utilisation de la même ressource par plusieurs jetons. Une fois ces contraintes établies, la résolution du problème de réalisabilité se fait en deux temps également : (a) la vérification de l'existence d'une execution temporelle (un processus) du réseau qui reflète le plan horaire et qui s'intègre bien dans le dépliage structurel, et ensuite (b) la vérification de la satisfiabilité des contraintes temporelles par les processus trouvés. Ceci permet de répondre à la première question. Pour répondre à la seconde question qui considère les probabilités, nous construisons un arbre dont les feuilles sont des classes d'états stochastiques transitoires [39]. Intuitivement, une classe d'état est une représentation symbolique d'un ensemble d'états accessibles par le tir d'une même transition à partir d'une autre classe. Une fois cet arbre construit, jusqu'à une profondeur donnée, il suffit de trouver un processus correspondant à un chemin dans l'arbre et s'exécutant avec une probabilité strictement positive.

Enfin, cette méthode peut naturellement être généralisée aux systèmes de transport urbain, ferroviaire et aérien (bus, trains et avions) ainsi que les lignes de production automatisées ou tout autre système analogue.

Contents

List of Figures

1 Introduction

Mobility is, more than ever, a key ingredient for the development of our modern societies. Many transportation modes are available in urban areas: tramways, buses, taxis, metros, and more recently light shared vehicles such as bicycles or scooters. In densely populated cities (e.g., New York, Paris, Tokyo...), the demand for public mobility options is exploding; this is due to many factors such as the increasing cost of fuel, and consequently of individual transportation [31], the concentration of population and jobs around large cities [29], and the separation of residential and working areas. Among the available transportation modes, public rail transport is one of the safest, most affordable, and most efficient ones. To answer a growing demand, stakeholders have to provide the physical infrastructures for urban rail systems (URS), but also means to operate them in the most efficient way. This work focuses particularly on rail *rapid transit* systems, i.e., rail systems that operate along their own right-of-way, with no access for other vehicles or for pedestrians [10].

URSs are part of the answer to the ever-increasing mobility needs in metropolitan areas. However, every rail system has bounded capacity due to physical and safety constraints. Indeed, trains have limited payloads, speeds and breaking capabilities, and have to be separated by safety distances in order to avoid collisions. These physical limitations impose an upper bound to the average time between successive train departures at stations, and consequently the number of passengers that can use metros, even when networks are operated at their maximal capacities. Beyond the capacity limitations, disturbances also limit the efficiency of URSs. Rail traffic is frequently subject to disrupting events. Examples of such events are failures of signaling systems, poor weather conditions (wet tracks, black ice, wind...) and passenger misconduct. This last cause of disturbance is prominent: straphangers holding doors is one of the main causes of delay [17]. Such disturbances divert the system from its ideal behavior and reduce its performance, and in the most severe situations cause durable instability. A challenge in URSs is hence to operate a network to achieve satisfactory capacity, especially at peak hours, even when incidents can delay trains. Efficiency of a railway system is often evaluated according to a quality standard, that defines measures for the quality of the provided service, and takes several criteria into account: delays, power consumption, passengers comfort, etc.

Traffic management

In order to cope with the effects of disturbances effects and to improve traffic, operators rely on the expertise of human dispatchers along with a set of *traffic management policies*. The term *regulation* is also frequently used to refer to these management policies. Each policy is an algorithm that has a specific objective, ranging from trains delays minimization to energy savings. The most common policies play with trains' dwell and running time profiles to minimize delays or equalize *headways*[1]. Basic policies simply impose a fixed interval for trains' departures from termini or reduce dwell times to recover from delays; while more elaborate ones aim at optimizing several criteria simultaneously. These algorithms are mostly constructed empirically to give an answer to specific traffic management problems, but, while they provide fairly good performance, their optimality is not proven. In addition to this, hardly any comparison between their performances is concluded.

In order to verify the well functioning of traffic management algorithms, rail solution providers such as ALSTOM often use very detailed simulation platforms. Such platforms are deployed in late phases of projects, and provide a faithful representation of the considered rail system: a quasi-exact copy of the mimicked system that only emulates trains and tracks. Testing traffic management algorithms is performed by running specific scenarios on the platform. This solution provides reliable results but only in late phases of a project, and at a significant cost. The nature of such a tool only allows for real-time simulations in which simulating a scenario takes the same time as in the reproduced real system. This does not allow for sampling techniques such as Monte-Carlo simulation to assess the general stochastic behavior of the system[2], as it would take an unreasonable amount of time to perform an exhaustive simulation campaign considering a significant set of randomly chosen disturbances. In a similar way, using simulation techniques with precise models and discretization of time (such as RAILSYS [58] or OPENTRACK [54]) is too costly to evaluate performance of a railway system and of its traffic management policies.

In parallel to the problem of performance evaluation of traffic management policies, operators frequently consider robustness issues. To simplify traffic management, a frequent technique is to adhere to a predetermined scenario called *schedule*. Schedules are optimal planning of departures and arrivals of trains, and are built to provide good performance, and also to resist minor delays that are unavoidable during a day of operation, i.e., allow for fast recovery using management policies. Many traffic management techniques try to adhere to a predetermined schedule. Similarly, many performance measures are evaluated by computing a distance between a log of a realized day w.r.t. to a predetermined schedule. It is hence important to be sure that a given schedule can be properly implemented by the running system. An important question in this context is called *realizability*, and asks whether a given schedule is achievable by a URS.

> The objective of this thesis is to provide effective tools and techniques to evaluate and compare traffic management policies, and to improve their efficiency. We address this challenge through the definition of formal models, that are sufficiently

[1] A headway is the difference in time between two successive passages from the same point of two trains running in the same direction.

[2] More details in Section 2.3.

accurate but yet allow for fast simulation of metros for long durations. The final objective is to compute metrics obtained by analyzing logs recorded during extensive simulation campaigns.

This thesis took place in the context of Project P22, an industrial collaboration between INRIA and ALSTOM that was focused on the analysis of traffic management algorithms.

Contribution: a concurrent stochastic and timed model for URSs

One of the main difficulties for the analysis of performance of URSs lies in finding a model that allows for efficient simulations to be able to perform Monte-Carlo experiments, i.e., execute a large number of random simulation runs in a reasonable amount of time. This calls for the use of models that abstract some details of the network and of rolling stock to speed up simulation. However, to obtain significant metrics, the considered models have to be accurate enough to reproduce the behavior of URSs, and in particular timing and randomness issues.

Trains movements in URSs can be decomposed as successions of running periods (a train is moving from one station to the next one) and dwell periods (the train is stopped at a station and passengers can board or alight from the train). A natural approach when considering cyber-physical systems is to use hybrid models. In this thesis, we follow a different approach. We consider that the train dynamics need not be finely depicted to obtain significant results on the performance of regulation algorithms, and model them with timed stochastic discrete-event systems (DES). We propose a framework to evaluate performance of traffic management algorithms for URSs using variants of stochastic time Petri nets (STPN). In particular, we show that STPNs are precise enough to model train movements and constraints linked to network topologies at an accurate enough abstraction level. We also show, through experimentation, that the concurrent nature of this model allows for fast simulations and, hence, for Monte-Carlo simulation campaigns to evaluate performance of URSs.

Petri nets [61] and their timed and stochastic variants have been widely used to model DESs. They have also been used to model toy examples in the railway community (BEHRMANN et al. [8] have, for instance, used such a model for the verification of railroad crossings), but rarely to model complete systems with dozens of trains. Similarly, fluid variants of Petri nets [60] have been proposed to model networks that convey quantities of objects. However, none of the existing models (up to our knowledge) allows to address at the same time durations, random disturbances, and adaptation mechanisms to ensure conformance to a desired schedule or performance. Indeed, the operation of a rail system normally follows a predefined schedule for departures and arrivals of trains. In the context of rail systems, this schedule is called a *timetable*. A timetable is an idealized vision of the desired and expected behavior of a system. It mainly consists of a series of dated departures and arrivals for a set of considered trains. A standard objective for train management is to follow as much as possible the schedule given by a timetable.

The models proposed in this thesis for URSs are stochastic variants of time Petri nets. These variants allow for the description of operations that take time such as trains trav-

eling from a station to another, dwelling at platforms or turnback maneuvers, and discrete events such as departures and arrivals. They also enable the representation of random delays due to disturbances. They implement timing and physical constraints such as safety distances between trains and speed limitations. More importantly, these models integrate regulation schemes, introduced as a particular controller that can impose execution of a transition or delay it (this is used to represent a particular speed or dwell time chosen by a regulation algorithm). These models are used to simulate the behavior of a real URS, provide an answer to the realizability question, and compute performance metrics for a URS with a particular regulation algorithm.

An orthogonal but yet important concern in the context of an industrial collaboration is modularity and reusability of the models. In this work, we adopt a modular modeling approach with three main components: 1. a module to animate trains and generate random delays, 2. one to represent the desired behavior (the timetable), and 3. one to describe the control mechanism that allows to recover from delays and get back to the desired behavior.

During a simulation these three components communicate to reproduce the behavior of a real controlled system. With this software architecture, it is rather straightforward to replace a regulation algorithm by another one, and run experiments to compare two traffic management policies. Simulation campaigns allow to record logs of days of operations, and then, to compute performance indicators, and assess the quality of a particular traffic management policy.

Outline

This document is organized as follows:

We first recall, in Chapter 2, some notions of probability theory. Probabilities are essential to model variations in durations of dwell and running times of trains. We also introduce a well known random sampling technique, namely the inverse transform sampling, that will be used to draw sample durations during the simulations. The second point addressed in this chapter is Monte-Carlo simulation. It is at the heart of our performance analysis method for URSs. We recall important theorems, namely the law of large numbers, that says that a sample mean converges toward the expected value of a random variable when the size of the sample grows. We also recall the *central limit theorem* that is a key ingredient to guarantee that this mean lies within a confidence interval for a chosen accuracy. These two results are essential to guarantee well foundedness of the Monte-Carlo simulation technique used in Chapter 7 to evaluate the performance of traffic management algorithms. We also recall, in this chapter, the Fourier–Motzkin variable elimination method. This technique is used to eliminate variables from a system of inequalities. It is used to prove that such a system has a solution and also to project these systems on a subset of their variables. This technique will be used later in Chapter 6 to guarantee existence of a solution for the implementation of a schedule.

Chapter 3 provides the necessary background on URSs. We introduce the notion of timetable that is a schedule for train departures and arrivals. It is a key ingredient in URSs as it is used both to control the system and as a reference to evaluate its performance. We then detail specificities of network topologies, and show two different signal-

ing systems for these topologies: the fixed-block (FB) and moving-block (MB) policies. In a FB policy, tracks are divided into portions (blocks) that can be occupied by at most one train at a time. In a MB policy, several trains can occupy the same track portion provided they respect certain security distances. We will see, in Chapter 5, that using one policy or the other has an impact on the type of models that we need to use. This chapter also defines traffic management and presents different techniques used by rail operators. We also formalize several standardized indicators used to address performance of URSs, the so called key performance indicators (KPI).

Then, Chapter 4 is a state of the art on techniques and models used in URSs. It mainly focuses on: (a) timetable design and management approaches, and (b) models for the representation and simulation of train movements in rail systems—or, more generally, for systems with routing and scheduling objectives. In particular, we consider time Petri nets, stochastic Petri nets, batches Petri nets, and queuing networks. We show that the features that need to be addressed to assess performance of traffic management techniques cannot be handled with these models which justifies the introduction of new models in Chapter 5.

In Chapter 5, we introduce two models developed to specify URSs with FB and MB policies. These models are variants of STPNs where distributions are associated with transitions and represent variability in dwell and transit times of trains. In the FB model, trains are represented as tokens. The model uses an elementary semantics to avoid train collision: a departure is forbidden if the next track portion is occupied. The MB model replaces tokens by train trajectories. In this model, train departures are constrained by the trajectories of other trains in the system (to satisfy safety headways). For these two models, we define a particular type of asymmetric distributions that model the fact that delays are more probable than advances. We equip the two models with a formal semantics that is at the heart of the simulation tool presented in Chapter 7.

Chapter 6 addresses the question of schedules realizability. Given a schedule for train operations, and a low-level description of the system for which the schedule is designed, we define a technique to check that the schedule is compatible with, at least, one run of the system, represented as an STPN. This question is called *boolean realizability*. It is solved using unfolding techniques: we show that boolean realizability can be brought back to the question of existence of an embedding of a schedule in a time process of an unfolding of the net representing the system. The difficulty here is that embedding is not a monotonous property. To cope with this problem, we show that this question can be answered on a bounded unfolding whose depth depends on the considered schedule and net. The second contribution of this chapter is *probabilistic realizability*. Indeed, boolean realizability does not address probabilities, and a realizable schedule might be realizable with a very low or null probability. We first define probabilistic realizability as the realizabbility of a schedule up to some imprecision with a sufficiently high probability. We answer this question using transient analysis techniques [39].

In Chapter 7, we show how the models introduced in Chapter 5 can be used to simulate operation of URSs under a specific traffic management algorithm and to evaluate their performance. We introduce the architecture of the SIMSTORS tool (for Simulator for Stochastic Regulated Systems) that we have developed during this project. We show results obtained on a real case study (namely the line 1 of Santiago's metro) with our tool. Beyond computation of performance indicators, we show how the importance of disturbances impacts the stability of the system: for small disturbances, simulation shows the

time needed for traffic management techniques to get back to a normal situation, while for important disturbances, the system can never recover from delays. This shows that our simulation tool can also be used to measure the robustness of a URS to disturbances.

The last chapter of this thesis concludes this work. We summarize and discuss the achieved results, list possible improvements of the techniques and tools used in this thesis, and draw lines for future research directions.

2 Preliminaries

This chapter introduces basic definitions of probability theory, Monte-Carlo simulation, and variable elimination that will be used later in the document.

2.1 Basic notions of probability theory

The models proposed in chapter 5 specify random durations (for dwell or running times). These duration will be captured by sampling values for *random variables*. A random variable X is a variable whose possible values are numerical outcomes of a random phenomenon. In other words, the value of a random variable is the result of a random experiment. Random variables can be *discrete* (they take values from a finite or infinite but countable domain), or *continuous*.

Definition 2.1 (random variable)
A *random variable* $X : \Omega \to E$ is a function that associates each element from a set of *outcomes* Ω with an element from a measurable space E (usually real numbers). The set of outcomes of X is called the *sample space*. ◊

An *event* is a subset of Ω. The result of a random experiment is denoted ω. For a discrete random variable X, the probability that X takes value k is the probability of the event $\{\omega : X(\omega) = k\}$, and is written $\mathbb{P}[X = k]$[1], or simply $p_X(k)$. The function p_X is called the *probability mass function* of X. The probability that X takes a value in a measurable set $S \subseteq E$ is written as $\mathbb{P}[X \in S]$.

Definition 2.2 (support)
The *support* of a random variable X is the set of values that X can take with a strictly positive probability. ◊

Definition 2.3 (probability density function)
A *probability density function (PDF)* of a (continuous) random variable X is a function

[1] Only applicable to *discrete* random variables.

associates to any given sample in the sample space the *relative likelihood* that the value of the random variable equals that sample. PDFs are usually denoted by $f_X(x)$. ◊

Probability Density Functions are used to associate probabilities to events depicting the fact that a random variable falls in a particular range of values. The probability for X to take values in a given interval $[a, b]$ is

$$\mathbb{P}[a \leq X \leq b] \triangleq \int_a^b f_X(x)\,\mathrm{d}x\,.$$

As a consequence, the probability of a single value is null. A graphical interpretation of PDFs is given by the area located between the horizontal axis and the density function between a and b. A PDF can only take nonnegative values and its integral over its domain is equal to 1.

Definition 2.4 (cumulative distribution function)
A *cumulative distribution function (CDF)* of a random variable X describes the probability for X to take a value less than or equal to x. A CDF for variable X is usually denoted by F_X, i.e. $F_X(x)$ is the probability that random variable X takes a value smaller than x. $F_X(x)$ can be expressed as a function of $f_X(x)$;

$$F_X(x) \triangleq \int_{-\infty}^{x} f_X(y)\,\mathrm{d}y\,.$$

Given two real numbers a and b, with $a \leq b$, we have:

$$\mathbb{P}[a \leq X \leq b] = F(b) - F(a).$$ ◊

Definition 2.5 (expected value)
The *expected value* $\mathbb{E}[X]$ of a given random variable X is the probability-weighted average of all possible values of X.

For discrete variables, the expected value is given by

$$\mathbb{E}[X] \triangleq \sum_{x_i 1 \Omega} p_X(x_i) x_i$$

For continuous variables, the expected value is given by

$$\mathbb{E}[X] \triangleq \int_{-\infty}^{+\infty} x f_X(x)\,\mathrm{d}x\,.$$

To simplify notations, we will often write μ_X instead of $\mathbb{E}[X]$. μ_X is also called the *theoretical mean* of X. ◊

Definition 2.6 (variance)
The *variance* $\mathbb{V}[X]$ of a random variable X is the expected value of the squared deviation from the mean μ_X. The variance of X quantifies the amount of dispersion of X around its expected value. For discrete variables, it is defined as

$$\mathbb{V}[X] \triangleq \sum_{x_i \in \Omega} p_X(x_i).(x_i - \mu)^2.$$

For continuous variables, it is defined as

$$\mathbb{V}[X] \triangleq \int_{-\infty}^{+\infty} (x-\mu)^2 f_X(x)\,\mathrm{d}x\,. \qquad \Diamond$$

Definition 2.7 (standard deviation)
The *standard deviation* of a random variable X quantifies the amount of dispersion of a set of data values. It is defined as the square root of its variance $\mathbb{V}[X]$. A low standard deviation indicates that the data points tend to be close to the expected value μ_X, while a high standard deviation indicates that the data points are spread out over a wider range of values It is frequently used to measure confidence in statistical conclusions. We denote the standard deviation of X by $\sigma_X \triangleq \sqrt{\mathbb{V}[X]}$. $\Diamond$

We can now show two frequent examples of continuous distributions, namely the normal distribution and the uniform distribution.

Definition 2.8 (normal distribution)
The *normal (or Gaussian) distribution* is a probability law that is suitable for the modeling of natural phenomenons which exact distributions are not known. The PDF of a normal distribution is defined according to two parameters μ and σ, where μ is the expected value and σ is the standard variation as follows:

$$g_{\mu,\sigma^2}(x) \triangleq \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

When a random variable X follows a normal law of parameters μ,σ, we say that X is *normally distributed* and write $X \sim \mathcal{N}(\mu,\sigma^2)$.

Furthermore, $\mathcal{N}(0,1)$ is called the *standard* normal law. $\Diamond$

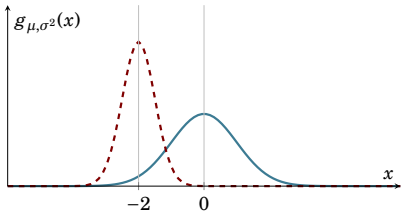


Figure 2.1: Gaussian distribution with parameters (0,1) (plain line) and (-2,0.5) (dashed line).

Definition 2.9 (uniform distribution)
The *uniform distribution* is a probability distribution that gives the same probability to all intervals of identical size in the distributions's support. A uniform distribution with support $[a,b]$ is denoted by $U(a,b)$, and its PDF is defined as

$$u_{a,b}(x) \triangleq \begin{cases} \frac{1}{b-a} & \text{if } x \in [a,b] \\ 0 & \text{otherwise.} \end{cases} \qquad \Diamond$$

Definition 2.10 (Weibull distribution)
The probability density function of a *Weibull distribution* is a law of the form

$$w_{\lambda,k}(x) \triangleq \begin{cases} \frac{k}{\lambda}.(\frac{x}{\lambda})^{k-1}.e^{-(x/\lambda)^k} & \text{if } x \geq 0 \\ 0 & \text{otherwise.} \end{cases} \qquad \Diamond$$

In this definition, k represents a shape parameter, and λ the scale parameter of the distribution. Weibull functions are interesting, as they allow for the modeling of asymmetric distributions. This is particularly important in Metro networks, where the probability of a delay is higher than the probability of an advance.

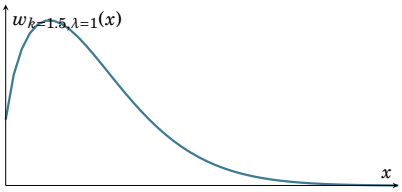


Figure 2.2: Weibull distribution with parameter $k = 1.5, \lambda = 1$.

2.2 Random sampling

Pseudo-random number sampling consists in generating pseudo-random numbers that are distributed according to a given probability distribution. Given a variable X with PDF $f_X(x)$ and CDF $F_X(x)$, one has to sample values in a way that is consistent with $f_X(x)$ and $F_X(x)$. Methods to sample values for a variable with a non-uniform distribution usually rely on pseudo-random generator producing numbers that are uniformly distributed. A well-known technique is the *inverse transform sampling method* (a.k.a. the *Smirnov method*). This technique can generate pseudo-random samples from a given CDF F_X. A prerequisite is that F_X is invertible, that is, for a cumulative distribution function, strictly increasing over its support. The Smirnov method is the following:

Algorithm 2.1: Inverse Transform Sampling

input : An invertible CDF F_X
output: A pseudo-random value x

1 sample a value u from the standard uniform distribution $U(0,1)$ in the interval $[0,1]$
2 **return** $x = F_X^{-1}(u)$

It is hence sufficient to have a pseudo-random generator for $U(0,1)$ to draw samples for a variable X with CDF F_X. One may wonder why such a simple technique works. We have $F_X^{-1}(u) = \inf\{x \mid F(x) \geq u\}$. Now, if U is a uniform random variable on $[0,1]$, then $F^{-1}(U)$ has F_X as CDF. This can be easily shown, as $\mathbb{P}(F_X^{-1}(U) \leq x) = \mathbb{P}(U \leq F_X(x))$ as F_X is monotonic. Now as U is uniform on $[0,1]$, we have that $\mathbb{P}(U \leq y) = y$ for every y in $[0,1]$. Hence we have $\mathbb{P}(F_X^{-1}(U) \leq x) = F(x)$.

The inverse transform sampling technique applies in many contexts. One can notice, for instance, that CDF for variables with normal laws, uniform laws, or Weibull distributions are strictly growing and invertible. With a pseudo random number generator for a random variable X, one can perform an arbitrary number of experiments, and approach via sampling the expected value μ_X. For a given variable X, let $S = x_1, \ldots x_n$ be a sequence of n *realizations* of X (i.e. a sequence of values sampled independently according to the distribution associated to X). The *sampled mean* $\overline{X}_n$ for sequence S is $\overline{X}_n \triangleq \frac{1}{n}\sum_{i=1}^{n} x_i$. According to the strong law of large numbers (SLLN), the sampled mean of a random variable converges to the expected value μ_X.

Theorem 2.1 (Strong law of large numbers)
Let X be a random variable, then the sample mean $\overline{X}_n$ converges almost surely *to μ_X, that is*

$$\mathbb{P}\left[\lim_{n\to+\infty} \overline{X}_n = \mu_X\right] = 1$$

◊

The strong law of large numbers is an important result: it guarantees that the average of the results obtained from a *large* number of experiments with variable X should be close to its expected value, and will tend to become closer as more experiments are performed.

2.3 Monte-Carlo simulation

Monte-Carlo (MC) simulation methods are a broad set of algorithms that rely on randomness to solve problems (that can be deterministic) that are generally *hard* to solve by means of analysis techniques. The principal idea is to run an extensive number of random experiments for the same process, and to derive conclusions from the average observed behavior.

When the expected value μ_X is unknown, it can be estimated by sampling a subset of outcomes $S \triangleq \{x_1, x_2, \ldots, x_n\}$ of size n, and calculating its mean value. We generally denote this value by $\overline{X}_S$ and call it the *sample mean* of S (or the *estimator* of μ_X). It is obtained by

$$\overline{X}_S \triangleq \frac{1}{n}\sum_{i=1}^{n} x_i\,.$$

Now, to estimate the standard deviation through the sample S, one can use the formula

$$\sigma_S \triangleq \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(x_i-\mu\right)^2};$$

but, since the expected value μ is unknown, we instead use

$$\widehat{\sigma}_S \triangleq \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(x_i-\overline{X}_S\right)^2}.$$

2.3.1 Central Limit Theorem

The *central limit theorem (CLT)* is the basis of the MC method: Monte-Carlo simulation can be used to estimate characteristics of physical system (e.g., the surface of a lake), or an output of a random process (e.g., chance of drawing *heads* following a coin toss). Monte-Carlo simulation estimate a mean from a set of values obtained by repeating the same stochastic experiment a large number of times. However, the estimated mean may not be reliable enough in a probabilistic context. The central limit theorem provides a notion of certainty to the estimated value obtained through simulation.

We recall that according to the law of large numbers (LLN), a given sample mean $\overline{X}_S$ converges to the expected value μ as n approaches $+\infty$. The (classical) *CLT* gives a description of the stochastic fluctuations around μ during this convergence. It states that the sampled mean of a large number of independent identically distributed variables $X_1,\ldots X_n$ approaches a normal distribution, regardless of the underlying distribution. More precisely:

Theorem 2.2 (central limit theorem)
Given $S \triangleq \{X_1, X_2, \ldots, X_n\}$ a series of independent identically distributed random variables of expected value μ and variance σ^2, and $\overline{X}_S \triangleq \frac{1}{n}\sum_{i=1}^{n} X_i$ an estimator of μ; then, the random variable

$$\frac{\overline{X}_S - \mu}{\sigma/\sqrt{n}} \tag{2.1}$$

converges in law toward the standard normal law $\mathcal{N}(0,1)$ when $n \longrightarrow +\infty$. ◊

The CLT is important: it implies that probabilistic and statistical methods that work on normal distributions also work on many other types of distributions. More interestingly, relating a distribution for a random variable X with the normal law allows for the definition of confidence intervals.

2.3.2 Confidence intervals

Let μ be an unknown expected value of a random variable in a stochastic system. Using MC simulation, one can calculate an interval $I \triangleq [\alpha, \beta]$, with a certain confidence that μ lies in this interval.

Before calculating I, the first step is to choose a desired level of confidence $c \in [0,1]$ with which the expected value is sought to lie in $[\alpha, \beta]$. A high c means a high level of confidence. We always have

$$\mathbb{P}\left[\mu \in [\alpha, \beta]\right] = c\,.$$

Now, following a number n of simulation runs, one will obtain a sample mean $\overline{X}_S$ and an estimated standard deviation $\widehat{\sigma}_S$. To compute the α and β, the endpoints of I, it is necessary to calculate an additional value γ called the *z-score*. This value gives an interval $[-\gamma, +\gamma]$ such that

$$\int_{-\gamma}^{+\gamma} g_{0,1}(x)\,\mathrm{d}x = c\,,$$

where $g_{0,1}$ is the standard normal function.

γ is obtained from the inverse of the standard normal distribution's CDF. Note, however, that there exists no explicit form of this CDF and, therefore, z-scores are rather found in precalculated tables, called the *z-tables*.

Once $\overline{X}_S$, $\widehat{\sigma}_S$, and γ known, the values of α and β are obtained by

$$\left[\begin{array}{l} \alpha \triangleq \overline{X}_S - \gamma\cdot(\widehat{\sigma}_S/\sqrt{n}) \\ \beta \triangleq \overline{X}_S + \gamma\cdot(\widehat{\sigma}_S/\sqrt{n}). \end{array}\right.$$

Remark 2.1
The statement

$$\overline{X}_S - \gamma \cdot (\widehat{\sigma}_S \,/\, \sqrt{n}) \leq \mu \leq \overline{X}_S + \gamma \cdot (\widehat{\sigma}_S \,/\, \sqrt{n})$$

is obtained by a trivial transformation[2], from

$$-\gamma \leq \frac{\overline{X}_S - \mu}{\sigma / \sqrt{n}} \leq +\gamma$$

that is a direct consequence of the CLT. ◊

This approach is illustrated by the example hereafter.

Example 2.1. Let us suppose that the desired level of confidence is $c = 0.95$. The z-score associated with c is then $\gamma \approx 1.96$.

According to Figure 2.3, that depicts a standard normal distribution, there is a 95% chance to see our standardized mean $\frac{\overline{X}_S - \mu}{\sigma/\sqrt{n}}$ in the interval $[-1.96, +1.96]$, that is

$$\mathbb{P}\left[-1.96 \leq \frac{\overline{X} - \mu}{\sigma/\sqrt{n}} \leq +1.96\right] \approx 0.95.$$

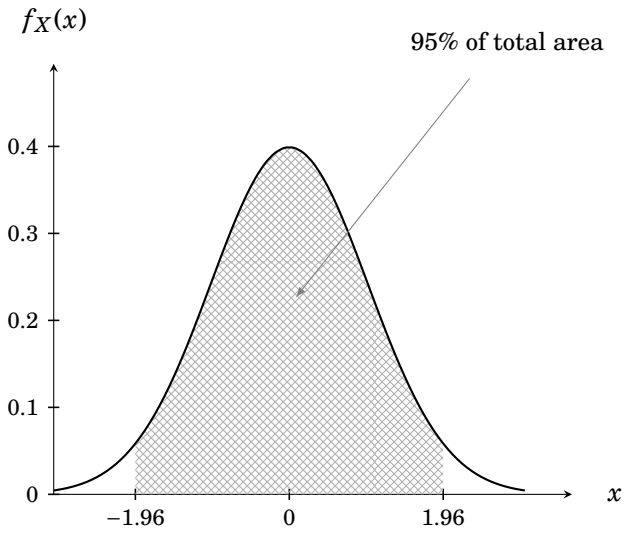


Figure 2.3: Area below the PDF of the standard normal distribution in $[-1.96, +1.96]$

With a simple transformation, we obtain the endpoints of the confidence interval

$$\left[\begin{array}{l} \alpha = \overline{X} - 1.96 \cdot (\sigma/\sqrt{n}) \\ \beta = \overline{X} + 1.96 \cdot (\sigma/\sqrt{n}). \end{array}\right. \tag{2.2}$$

Now suppose that our sequence of samples is $S = \{8, 10, 5, 10, 8, 9, 7, 11, 13, 10, 2, 10, 10\}$. It is of size $n = 10$ and produces a sample mean $\overline{X}_S \approx 8.69$. Supposing also that σ is

[2] n.b., we suppose here that the standard deviation σ is also unknown and replace it with its estimation $\widehat{\sigma}_S$.

unknown, computing an estimate yields $\widehat{\sigma} \approx 2.81$. Using Formula (2.2), we obtain

$$\mathbb{P}\left[7.16 \leq \mu \leq 10.22\right] \approx 0.95.$$

In a nutshell, MC simulation is performed through the following steps:

1. generate n samples to obtain a sequence $S \triangleq \{X_1, X_2, \ldots, X_n\}$,
2. compute the sample mean $\overline{X}_S$ and the estimated standard deviation $\widehat{\sigma}_S$,
3. choose a confidence level c with which we want the expected value μ to be in a confidence interval $I = [\alpha, \beta]$,
4. find the corresponding γ in a z-table or by means of a scientific computing software (e.g., R or MATLAB),
5. deduce the endpoints α and β of the confidence interval according to Formula (2.2).

We can finally say that the outcome is expected to lie within the interval $[\alpha, \beta]$ with $(c \cdot 100)\%$ certainty.

2.4 Fourier–Motzkin elimination

In Chapter 6, we will work with symbolic descriptions of durations, depicted by sets of linear constraints. To manipulate constraints in a symbolic way, elimination is a key operation. A constraint $A(x_1, \ldots, x_r)$ over a set of real-valued variables $V = \{x_1, \ldots x_r\}$ is a set of linear inequalities. Assuming that $A(x_1, \ldots x_r)$ contains n inequalities, it can be depicted as a conjunction of the form

$$A(x_1, \ldots, x_n) = \begin{cases} & a_{1,1}.x_1 + a_{2,1}.x_2 + \cdots + a_{r,1}.x_r & \leq & b_1 \\ \wedge & a_{1,2}.x_1 + a_{2,2}.x_2 + \cdots + a_{r,2}.x_r & \leq & b_n \\ . & & & \\ . & & & \\ \wedge & a_{1,n}.x_1 + a_{2,n}.x_2 + \cdots + a_{r,n}.x_r & \leq & b_n \end{cases}$$

where $a_{i,j}$'s and $b'_j s$ are rational values. $A(x_1, \ldots x_r)$ is satisfiable if one can find a valuation for $x_1, \ldots, x_r$ such that replacing each variable x_i by its value yields a tautology for each inequality. Elimination of a variable x_r from a system $A(x_1, \ldots x_r)$ with n inequalities consists in computing another *equivalent* constraint $B(x_1, \ldots x_{r-1})$, i.e. such that $A(x_1, \ldots x_r)$ and $B(x_1, \ldots x_{r-1})$ have the same solutions over the remaining variables. In particular, $A(x_1, \ldots x_r)$ is satisfiable iff $B(x_1, \ldots x_{r-1})$ is satisfiable. Elimination is hence a key technique to decide if a system of inequalities has a solution: one can eliminate all variables and check that the result obtained after elimination is a tautology.

The Fourier–Motzkin elimination (FME) method is an algorithm for eliminating variables from a system of linear inequalities. Without loss of generality, we will assume that we eliminate variable x_r. The method eliminates x_r with the following steps:

Partition The set of inequalities in $A(x_1,\ldots x_r)$ can be partitioned into three sets of inequalities $A^+(x_1,\ldots x_r)$, $A^-(x_1,\ldots x_r)$, and $A^\varnothing(x_1,\ldots x_{r-1})$.

$A^+(x_1,\ldots x_r)$ is the set of inequalities in $A(x_1,\ldots x_r)$ that are of the form

$$x_r \leq b_j - \sum_{k=1}^{r-1} a_{k,j}.x_k$$

$A^-(x_1,\ldots x_r)$ is the set of inequalities in $A(x_1,\ldots x_r)$ that are of the form

$$-x_r \leq b_j - \sum_{k=1}^{r-1} a_{k,j}.x_k$$

and $A^\varnothing(x_1,\ldots x_{r-1})$ is the set of inequalities that do not contain x_r, i.e. of the form $a_{1,j}.x_1 + a_{r-1,j}.x_{r-1} + 0.x_r \leq b_j$.

Clearly, a valuation is a solution for $A(x_1,\ldots x_r)$ iff it is a solution for $A^+(x_1,\ldots x_r)$, $A^-(x_1,\ldots x_r)$ and $A^\varnothing(x_1,\ldots x_{r-1})$. Let m be the number on inequalities in $A^+(x_1\ldots,x_r)$, q be the number of inequalities in $A^-(x_1\ldots,x_r)$. $A^+(x_1\ldots,x_n)$ defines a constraint of the form $\bigwedge_{i=1..m} x_r \leq A_i^+(x_1,\ldots,x_{r-1})$ where A_i^+ is an expression of the form $b_i - \sum_{k=1}^{r-1} a_{i,k}.x_k$. Similarly $A^-(x_1\ldots,x_n)$ defines a constraint of the form $\bigwedge_{i=1..q} x_r \geq A_i^-(x_1,\ldots,x_{r-1})$ where A_i^- is an expression of the form $b_i - \sum_{k=1}^{r-1} a_{i,k}.x_k$.

The original system can be written as

$$A^\varnothing \wedge \max\left(A_1^-(x_1,\ldots,x_{r-1}),\ldots,A_q^-(x_1,\ldots,x_{r-1})\right) \leq x_r \leq \min\left(A_1^+(x_1,\ldots,x_{r-1}),\ldots,A_m^+(x_1,\ldots,x_{r-1})\right)$$

Obviously, there exists a valuation for $x_1,\ldots,x_r$ iff there exists a valuation for $x_1,\ldots x_{r-1}$ satisfying the following constraint:

$$A^\varnothing \wedge \max\left(A_1^-(x_1,\ldots,x_{r-1}),\ldots,A_q^-(x_1,\ldots,x_{r-1})\right) \leq \min\left(A_1^+(x_1,\ldots,x_{r-1}),\ldots,A_m^+(x_1,\ldots,x_{r-1})\right)$$

min and max can then be eliminated by considering conjunctions of pairwise comparisons of A_i^-'s and A_j^+'s, leading to a constraint of the form:

$$B(x_1,\ldots x_{r-1}) = A^\varnothing \wedge \bigwedge_{i=1\ldots q, j=1\ldots m} A_i^-(x_1,\ldots x_{r-1}) \leq A_j^+(x_1,\ldots x_{r-1})$$

That is we obtain, in addition to $A^\varnothing$, conjunctions of inequalities of the form $b_i - a_{1,i}.x_1 - a_{2,i}.x_2 - \ldots a_{r-1,i}.x_{r-1} \leq b_j - a_{1,j}.x_1 - a_{2,j} - \ldots a_{r-1,j}.x_{r-1}$ That can be rewritten as $(a_{1,j} - a_{1,i}).x_1 + (a_{2,j} - a_{2,i}).x_2 + \cdots + (a_{r-1,j} - a_{r-1,i}).x_{r-1} \leq b_j - b_i$

which is still a system of linear inequalities, but over $x_1,\ldots x_{r-1}$.

Example 2.2. Let $A(x_1,x_2,x_3)$ be the following system of linear inequalities

$$A(x_1,x_2,x_3) \triangleq \begin{cases} \alpha_1 & \leq x_1 & \leq \beta_1 \\ \alpha_2 & \leq x_2 & \leq \beta_2 \\ \alpha_3 & \leq x_3 & \leq \beta_3 \\ -\gamma_{21} & \leq x_1 - x_2 & \leq +\gamma_{12} \\ -\gamma_{31} & \leq x_1 - x_3 & \leq +\gamma_{13} \\ -\gamma_{32} & \leq x_2 - x_3 & \leq +\gamma_{23} \end{cases}$$

and let x_1 be the variable to eliminate.

The first step consists in identifying $A^{\varnothing}(x_2,x_3)$, the subsystem of inequalities in which x_1 does not appear, as follows

$$A(x_1,x_2,x_3) = \begin{cases} \alpha_1 & \leq x_1 & \leq \beta_1 \\ -\gamma_{21} & \leq x_1 - x_2 & \leq \gamma_{12} \\ -\gamma_{31} & \leq x_1 - x_3 & \leq \gamma_{13} \\ \alpha_2 & \leq x_2 & \leq \beta_2 \\ \alpha_3 & \leq x_3 & \leq \beta_3 \\ -\gamma_{32} & \leq x_2 - x_3 & \leq \gamma_{23} \end{cases} \triangleq A^{\varnothing}(x_2,x_3)$$

We then isolate x_1 by rewriting the remaining inequalities, as follows

$$A(x_1,x_2,x_3) = \begin{cases} \alpha_1 & \leq x_1 & \leq +\beta_1 \\ x_2 - \gamma_{21} & \leq x_1 & \leq +x_2 + \gamma_{12} \\ x_3 - \gamma_{31} & \leq x_1 & \leq +x_3 + \gamma_{13} \\ A^{\varnothing}(x_2,x_3) & & \end{cases}$$

One can easily see that a solution for this system is a valuation for x_1,x_2,x_3 that satisfies $A^{\varnothing}$, and where x_1 is greater than the maximum value of its left bounds and smaller than the minimum value of its right bounds in the system of inequalities. Hence, we have :

$$A(x_1,x_2,x_3) = A^{\varnothing} \wedge \max\left(\alpha_1, x_2 - \gamma_{21}, x_3 - \gamma_{31}\right) \leq x_1 \leq \min\left(\beta_1, x_2 + \gamma_{12}, x_3 + \gamma_{13}\right)$$

Then, after elimination of x_1, one just compares the lower and upper bounds for x_1 to obtain

$$B(x_2,x_3) = A^{\varnothing}(x_2,x_3) \wedge \underbrace{\max\left(\alpha_1, x_2 - \gamma_{21}, x_3 - \gamma_{31}\right) \leq \min\left(\beta_1, x_2 + \gamma_{12}, x_3 + \gamma_{13}\right)}_{\theta}$$

Note that the inequality θ in this example is a system of 3×3 inequalities. Last, we

obtain :

$$B(x_2,x_3) = A^{\varnothing}(x_2,x_3) \wedge \begin{cases} \alpha_1 & \leq \beta_1 \\ x_2 - \gamma_{21} & \leq \beta_1 \\ x_3 - \gamma_{31} & \leq \beta_1 \\ \alpha_1 & \leq x_2 + \gamma_{12} \\ x_2 - \gamma_{21} & \leq x_2 + \gamma_{12} \\ x_3 - \gamma_{31} & \leq x_2 + \gamma_{12} \\ \alpha_1 & \leq x_3 + \gamma_{13} \\ x_2 - \gamma_{21} & \leq x_3 + \gamma_{13} \\ x_3 - \gamma_{31} & \leq x_3 + \gamma_{13} \end{cases}$$

and finally

$$B(x_2,x_3) = \begin{cases} \alpha_2 & \leq x_2 & \leq \beta_2 \\ \alpha_1 + \gamma_{12} & \leq x_2 & \leq \beta_1 + \gamma_{21} \\ \alpha_3 & \leq x_3 & \leq \beta_3 \\ \alpha_1 + \gamma_{13} & \leq x_3 & \leq \beta_1 + \gamma_{31} \\ -\gamma_{32} & \leq x_2 - x_3 & \leq \gamma_{23} \\ -\gamma_{12} - \gamma_{31} & \leq x_2 - x_3 & \leq \gamma_{13} + \gamma_{21} \\ & \alpha_1 - \beta_1 & \leq 0 \\ 0 & \leq \gamma_{12} + \gamma_{21} & \\ 0 & \leq \gamma_{13} + \gamma_{31}\,. & \end{cases}$$

Let $m = |A^+(x_1,\ldots x_r)|$ and $q = |A^-(x_1,\ldots x_r)|$. Elimination of a variable x_r from $A(x_1,\ldots x_r)$ produces a new system of inequalities of maximum size $m \times q + |A^{\varnothing}(x_1,\ldots x_{r-1})|$. The worst case for elimination of x_r from a set of n inequalities occurs when $A^{\varnothing}(x_1,\ldots x_{r-1}) = \emptyset$ and $m = q = n/2$. In this case, elimination produces $(n/2)^2$ inequalities. Thus, for r consecutive eliminations, the size of the obtained system is in $O(4 \cdot (n/4)^{2^r})$, i.e. is doubly exponential.

3

Context: Urban rail systems

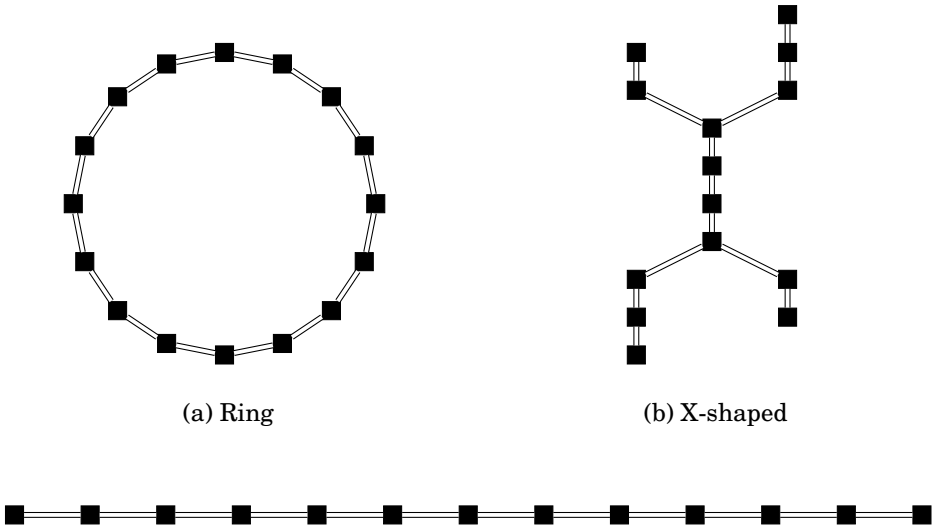
The objective of this chapter is to detail the context of the thesis, that is, management of urban train systems. We introduce the notions of network topologies, timetables, traffic management policies, performance indicators, and signaling systems. To facilitate reading of the rest of the document, we end this chapter with a glossary of terms frequently used in urban trains management.

3.1 Network topologies

Seen from a high-level perspective, urban rail systems (URSs) are composed of fleets of trains transporting passengers through a network. Each network is composed of stations and interstations arranged into a given topology. This topology is defined by the configuration of tracks (interstations) and trains' stopping points (stations). Topologies can vary from simple lines or circles to more complex configurations such as star-shaped topologies or grids.

Figure 3.1 gives examples of URS topologies. Subfigure 3.1c is a single line. This type of topology can be met for instance in Algiers and Rennes Metros. Subfigure 3.1a is a ring topology. This type of topology is used, for instance in the Glasgow Subway, which is a 10.5 km long bi-directional circle. Subfigure 3.1b is an X-shaped topology. Such a topology can be seen on the RER B line in Paris. This topology also appears in more complex systems where common track portions are shared by several lines (e.g., Brussels and Amsterdam Metro). Note that topologies show how station are interconnected, but that connections are most often bidirectional, and usually physically implemented by pairs of parallel tracks, allowing trains to run in both directions.

Traffic management in simple topologies where commercial lines are well separated is usually performed by considering each line independently. However, even the simplest lines contain *junctions* and *forks* that allow for insertions and removals of trains from a line, for turnback maneuvers, and for trains' overtakings. In terms of configuration of tracks, junctions and forks are Y-shaped track portions. What differentiates the former



(a) Ring

(b) X-shaped

(c) Line

Figure 3.1: Examples of topologies of urban rail networks

from the latter is the direction of trains movements. A junction, as illustrated in Figure 3.2, is the convergence of trains toward the same point; while a fork is a divergence.

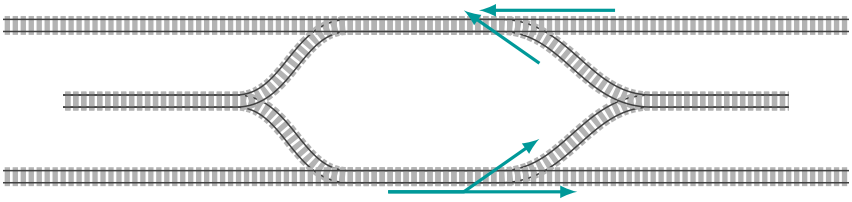


Figure 3.2: A junction (top) and a fork (bottom)

During traffic management, junctions and forks call for a particular care, and impose constraints in the scheduling of trains' order and times of passage. The arrivals of trains at junctions should be planned to fulfill headway constraints. Furthermore, junctions and forks use shunts that must be properly positioned *before* trains passage. Hence, when two trains arrive at a fork and do not follow the same direction, the time that elapses between the two trains must be sufficient to position the shunt properly. Similarly, trains shall not be too close from its predecessor. The rule is to maintain a security distance to avoid collision. If the distance between two trains falls below this safety threshold, the second train has to brake. A proper scheduling prevents the use of emergency braking and collisions that could be caused by two trains converging at the same junction. Tracks and trains are equipped with signaling systems that allow for the localization of trains and provide the occupation status of sections of the network. Human operators in trains usually rely on fixed signals: traffic lights with several colors: green, yellow, and red. Red means that the position immediately ahead is occupied, and the driver should brake immediately. Yellow means that a train is ahead, but at reasonable distance, and that the driver should reduce his speed. Green color means that the track ahead is free for a sufficiently long distance, and that the driver can run his train at commercial speed. Driverless trains use more advanced systems, but with the same spirit, meant to guarantee safety of passengers. These signaling and protection mechanisms ensure that trains

are sufficiently spaced and run at appropriate speeds to avoid collisions and unnecessary emergency brakings.

An operation day of a URS consists in performing commercial train services. Each train is associated with a service that defines its complete path in the network, along with stopping and speed time profiles. Services can be seen as a chain of successive trips where each trip starts from an origin terminus and ends in a destination terminus. A pair of sucessive trips in two opposite travel directions is usually linked with a turnback maneuver. During each trip, trains serve a number of stations at which they have to stop for the boarding and alighting of passengers. These stopping times are called *dwell* times and should normally be equal to the durations specified in the service associated with the considered train. However, during the operation day and especially at peak hours, these dwell times may not be respected: a heavy load of passengers increases boarding and alighting times, and sometimes doors closing are delayed. Some scenarios even include passengers deliberately holding trains' doors, a source of significant service degradation. On the other hand, other disturbances may occur at interstation level and lead to a forced change in speed profiles. Disturbances in interstations can be due to small incidents on tracks, but also to bad weather conditions.

3.2 Timetables

To facilitate traffic management, operators use predefined schedules that specify arrivals and departure dates of trains, but also ordering of these trains at forks and junctions. This reference schedule is called a *reference timetable*. Each commercial line has several reference timetables corresponding to an expected scenario: normal work day, holiday, weekend... When disturbances occur, operators use traffic management techniques to bring the system back to a reference behavior. In the rest of the paper, we will designate by Σ_{st} the set of stations appearing in the network, and by Σ_{tr} the set of operated trains.

Definition 3.1 (reference timetable)
A reference timetable can be formalized as a tuple $\mathcal{T} \triangleq \langle N, \rightarrow, d, \text{TP}, \text{TR}, \text{ST} \rangle$, where

- $\langle N, \rightarrow \rangle$ is a directed acyclic graph (DAG) where
  - $N \triangleq \{n_1, n_2, \ldots, n_q\}$ is a set of events, and
  - $\rightarrow \subseteq N^2$ is a precedence relation.

  For all pair of events, $\forall n_i, n_j \in \rightarrow$ means n_i has to occur before n_j.
  We usually simply write $n_i \rightarrow n_j$.
- $d : N \rightarrow \mathbb{R}_{\geq 0}$ assigns, to each event $n_i \in N$, a reference date $d_i \triangleq d(n_i)$;
- $\text{TP} : N \rightarrow \{\text{DEP}, \text{ARR}\}$ assigns, to each event, a type: departure (DEP) or arrival (ARR);
- $\text{TR} : N \rightarrow \Sigma_{tr}$ assigns, to each event, a train *tr* from the set of operated train Σ_{tr}; and
- $\text{ST} : N \rightarrow \Sigma_{st}$ assigns, to each event, a station *st* from the set of stations Σ_{st}. ◊

During operation, timetable-based traffic management techniques try to keep the system running as close as possible to the reference timetable. However, when the system is exposed to a significant level of disturbance, the reference timetable may become unachievable, i.e., we know in advance that events in a subset $N' \subseteq N$ cannot and will not occur at their reference dates. To illustrate this, let us, for example, suppose that a train

tr_1 was planned to arrive at station st_5 at date 1000, but that it is delayed by 35 seconds and arrives at its next station st_5 at a date $\dot{d}_5^a = 1000 + 35$. Let us also suppose that train tr_1 was scheduled to dwell for 18 seconds at station level before departing at date $d_5^d = 1000 + 18$. We can clearly see here that the planned departure date cannot be achieved anymore, even if the dwell time is reduced, as $\dot{d}_5^a > d_5^d$. For this reason, instead of aiming at realizing the reference timetable all along the operation day, operators use a copy of this reference timetable and update it. Updating a timetable means changing the occurrence dates of departures and arrivals so that the new dates obtained are compatible with the current delays of trains. Updating a timetable may also imply rescheduling of trains passages at forks and junctions, and optimization of some quality objectives. The updated timetable is a schedule that is achievable from the current state of the network, provided the system is not subject to new disturbances. This updated timetable differs from the reference timetable. It is modified online during the operation day. To keep track of the current state of the network, of the dates for the arrivals and departure that have already occurred, and of the expected ones, we use a timetable structure with more information called the *active timetable*.

Definition 3.2 (active timetable)
An active timetable can be defined as a tuple $\tilde{\mathcal{T}} \triangleq \langle N, \rightarrow, \dot{d}, \widehat{d}, \mathrm{TP}, \mathrm{TR}, \mathrm{ST}, \mathrm{EX} \rangle$ where

- N, $\rightarrow$, TP, TR, and ST keep the same definition as in Definition 3.1;
- $\mathrm{EX} : N \rightarrow \{\top, \bot\}$ indicates whether each event $n_i \in N$ is *executed* ($\top$) or not ($\bot$);
- $\dot{d} : \mathrm{EX}^{-1}(\{\top\}) \rightarrow \mathbb{R}_{\geq 0}$ assigns, to each executed event, an *effective occurrence date*; and
- $\widehat{d} : \mathrm{EX}^{-1}(\{\bot\}) \rightarrow \mathbb{R}_{\geq 0}$ assigns, to each nonexecuted event, a reference date $\widehat{d}(n_i)$. ◊

The active timetable can be divided into two parts:

- the *executed timetable* $\dot{\mathcal{T}}$, a restriction of components of $\tilde{\mathcal{T}}$ to executed events, and
- the new (realizable) timetable $\widehat{\mathcal{T}}$, a restriction to nonexecuted events.

At the end of an operation day, the executed timetable $\dot{\mathcal{T}}$ gives the dates of occurrence of all events, and can be used a a log to derive daily, weekly or monthly statistics.

Active timetables are not simply used to record realized dates of arrivals and departures of trains. They are used to give orders to trains. To realize the yet unexecuted part of an active timetable, trains have to leave at the prescribed dates, and run at speeds that allow them to arrive at the expected date at the next station.

In addition to the reference or active timetable, events occurrence dates are subject to constraint, that originate from maximal speeds and minimal dwell time. As a train has a maximal speed, if a node n represents a departure and another node n' the next arrival of the same train, then the time that must elapse between the two events is the minimal time that a train takes to go from one station to the other. If n represents an arrival at a station, and n' the next departure, then the time that must elapse between the occurrence of n and n' is the minimal dwell time specified for the network. Last if n represents an arrival of a train, and n' the arrival Of the next train at the same position in the network, then the time that must elapse between these two events is the minimal headway for the network. We assume that in, addition to the reference and active timetable, a partial map $\mathbb{C} : N \times N$ specifying minimal time that must elapse between successor events is provided.

Remark 3.1
The set of events N can be split into $N^d \triangleq \{n_1^d, n_2^d, \ldots, n_{qd}^d\}$ and $N^a \triangleq \{n_1^a, n_2^a, \ldots, n_{qa}^a\}$ for

the subset of departures and arrivals, respectively. These subsets can themselves be decomposed into more subsets to regroup events by train, by station, or both. ◊

Remark 3.2
For each event n, or each event's date $d(n)$, $\dot{d}(n)$, etc., we will often simplify notations and associate a subscript or a superscript that indicates its *characteristics*, i.e., its corresponding type, train index, station index, etc. For example, depending on context, we will write $\dot{d}^{d}_{i,t_j,s_k}$ to denote the occurrence date $\dot{d}(n)$ of a departure event n with index i in N, and corresponding to a train of index t_j in Σ_{tr} and a station indexed by s_k in Σ_{st}. On the other hand, when the context is clear, we will drop these indexes and simply write $\dot{d}$ or $\dot{d}^{d}$. ◊

Safety and deviation compensation

One of the main concerns during the construction of a timetable is safety: trains must run at appropriate speeds and respect *safety headways*. These headways are the *minimum* times that can elapse between the passage of two successive trains from the same point, in such a way that if the leading train stops, the trailing train will be able to break and stop in time, avoiding a head-to-tail collision. In commercial timetables, trains are always separated, at all times, by at least safety headways plus certain margins. These margins serve safety, but are also used to recover from delays by selecting faster speed profiles if needed. Nevertheless, playing on speeds as a means of *deviation compensation* (i.e., delay or advance recovery) is usually not the most efficient approach to recover from a delay, as trains usually recover one second between two stations by increasing their speed. On the other hand, dwell times can be significantly reduced or increased, and offer more interesting recovery solutions than reduction of running times. Reducing a dwell time when a train is late allows for departures at dates that are as close as possible to the one prescribed by the timetable. These mechanisms that play on dwell and running times to recover from a primary delay are one example of a broader set of traffic management techniques that allow the system to get back to a desired operation state. These techniques can be defined with rules, as functions of the current state of the system, with algorithms, etc. Their objective is to achieve the best possible service by providing adapted dwell or speed orders. This quality of service is usually quantified by KPIs that give a measure of punctuality of trains, regularity of service, or other passenger-centric indicators such as density of passengers or regularity expressed in terms of passenger waiting times [5].

3.3 Traffic management

Traffic management uses of a set of different techniques. The applied techniques depend on the operation day scenario, on the line, and also on decisions taken by operators. Each technique has its own purposes and aims at optimizing certain metrics. It can be seen as a function REG that takes as input an active timetable $\tilde{\mathcal{T}}$ and returns an updated active timetable $\tilde{\mathcal{T}}' = \text{REG}(\tilde{\mathcal{T}})$.

Traffic management techniques can be classified into two main categories: *terminus* and *mainline*. Terminus traffic management techniques set the dates of train departures from termini during insertions into the commercial line or turnback maneuvers. Mainline techniques, on the other hand, handle scheduling of train departures and arrivals at stations. These two types of techniques are operated jointly and in parallel. Each category

comprises a set of techniques (also called *policies*). We present hereafter some of the techniques used in the industry.

3.3.1 Terminus policy

Operator policy

In this policy, a human operator is in charge of setting departure dates of trains from termini. It does not rely on any automation mechanism. A departure date from a given terminus is usually function of (a) the preceding arrival date at the terminus and (b) an interval of possible values for the dwell time at the terminus. The operator chooses a value from this interval to compensate eventual deviations (w.r.t. the planned dates) and ensure a departure in time. If the deviation is too important, it will inevitably propagate to the next station.

Interval policy

Instead of aiming for a respect of the departure dates planned in a reference timetable, and especially when these dates becomes unfeasible, operators may want to simply provide a regular service in terms of intervals between departures. To do so, two approaches are possible.

The first approach consists in setting a departure date for a train from a terminus, and then, setting all following departures based on that date and a given constant interval δ. Given an i^{th} departure date $\widehat{d}_{i,j}$ from a terminus j, the k^{th} next departure from the same terminus is simply computed by

$$\widehat{d}'_{i+k,j} = \widehat{d}_{i,j} + k \cdot \delta \,.$$

This approach takes into account the theoretical departures dates of trains without considering eventual deviations. In doing so, the constant interval δ is not guaranteed. This policy is generally used in degraded situations.

The second approach computes the departures dates of trains from the observed effective departure dates of the preceding train and a given desired interval δ. Given the i^{th} departure date $\dot{d}_{i,j}$ from a terminus j, the next departure date is computed by

$$\widehat{d}'_{i+1,j} = \dot{d}_{i,j} + \delta \,,$$

where $\dot{d}_{i,j}$ is the date of the observed effective i^{th} departure (which can be different from the reference date $\widehat{d}_{i,j}$). This method ensures a certain regularity at terminus level but may result in significant deviations with respect to the original reference timetable, as it propagates delays.

3.3.2 Mainline policies

No action policy

In some scenarios, and especially when a rail system has already deviated too much from its planned behavior, it is possible to choose simply not to apply any traffic management action. This is practically done through selecting a *no action* policy in which trains keep

their nominal dwell times and proceed without trying to recover from delays. This is a degraded mode where delay (and advance) is only observed and logged. This leads to a shift in timetable events' dates according to the observed deviations. No particular action is taken, but this is considered as a valid traffic management technique and it is one of the actual possible choices an operator can take. The active timetable has to be recomputed, nevertheless, to update the predicted dates of future events.

Schedule policy

Timetable or schedule policy is one of the most natural and commonly used traffic management techniques. It uses a reference timetable and aims to get back to it, as soon as possible, whenever a deviation (a delay or an advance) occurs. At the beginning of an operation day, the active timetable is a copy of this reference timetable. The active timetable is updated during the operation day to record deviations, and return to the reference timetable. The deviation compensation mechanism plays with trains' dwell times at stations and running times between them. For example, when a train arrives late at a station, its dwell time is reduced. If the delay is small, this reduction can be sufficient to let the train leave the station at that date scheduled in the reference timetable. However, networks impose minimal (and maximal) dwell times to let sufficient time for passengers to board or alight. Hence, reducing a dwell time to its minimal value may not be sufficient to recover from a primary delay. In this case, the delay recovery can be performed progressively through the reduction of the dwell times for several successive stations. Sometimes, running times are also used as a recovery mechanism and are reduced by selecting faster train speed profiles. One should note, however, that the delay compensation using running times is usually less efficient than with dwell times, due to the limitations in trains' speeds and to the short distances between stations.

The constraints on minimum and maximum dwell and running times are defined as intervals containing the nominal values for these operations. These intervals are defined as $[\underline{\Delta}(st), \overline{\Delta}(st)]$ for each station st by the traffic management policy. They are generally not constant; they change over the course of the operation day. Given $\Delta(st)$ a nominal dwell or running time value for station st, the differences $\underline{\eta}(st) \triangleq \underline{\Delta}(st) - \Delta(st)$ and $\overline{\eta}(st) \triangleq \overline{\Delta}(st) - \Delta(st)$ between the endpoints of the interval and the nominal value for an operation are called *deviation compensation time margins*. Time margins are used in this policy, and the bigger they are, the faster the network will absorb deviations and get back to a reference timetable.

Example 3.1
To illustrate this traffic management policy, let us suppose that, in an active timetable $\tilde{\mathcal{T}}$, a train tr expected to arrive at a station st at date $\widehat{d}^a = 100$ is delayed by $\omega = 8$ seconds, and arrives at $\dot{d}^a = 100 + 8$. Suppose also that the departure from station st was scheduled at $\widehat{d}^d = 100 + 20$, with a nominal dwell time $dt = 20$ and time margins $[-10, +60]$. This policy will then choose a new dwell time $dt' = 20 - 8$ seconds, and in doing so, the train will be able to depart in time. If the delay is bigger, e.g., $\omega = 12$, and given that the minimum allowed dwell time is $\underline{\Delta}(st) = 20 - 10$ seconds, the train would depart at a new date $\widehat{d}^{d\prime} = (100 + 12) + 10$ with a delay of 2 seconds w.r.t. the scheduled date $\widehat{d}^d$. ◊

Let us now show how to recompute earliest dates of events in an active timetable by playing with dwell times. Let x be a node that is delayed, i.e., such that $d(x) < \dot{d}(x)$. The idea is to compute all successors of x, i.e., nodes whose occurrence date can be impacted by the delay, and propagate the minimal possible delay among these nodes by executing them

at the earliest possible date occurring after their planned execution date. To compute the new date of an event, we first need to know the date of all its predecessors. For this, we affect new dates according to a topological order, that is a total order on nodes, that guarantees that, if $x \rightarrow y$, then x occurs before y in the topological order. We do not detail topological ordering algorithm, and refer readers to [43], p258. Algorithm 3.1 below shows how modifications can be easily propagated in an active timetable.

Algorithm 3.1: Rescheduling a timetable after a primary delay

input : A timetable $\tilde{\mathcal{T}} = (N, \rightarrow, d, \hat{d}, TP, TR, ST, EX)$,
a node x, its expected date $\hat{d}(x)$,
its actual realization date $\dot{d}(x)$
output: an updated timetable $\tilde{\mathcal{T}}'$

`/* Build successors of` x `*/`

1 $Succ(x) = \{x\}$
2 **while** $Succ' \neq Succ(x)$ **do**
3 $\quad Succ(x) = Succ'(x)$
4 $\quad Succ'(x) = Succ(x) \cup \{y \mid \exists n \in Succ(x) \wedge n \rightarrow y\}$
5 Build a topological order $n_1 \ldots n_k$ on $Succ(x)$
6 **for** $i \in 1..k$ **do**
$\quad$ `/* update the occurrence date of` n_i `*/`
7 $\quad dpred = \max\{d(x) + \mathbb{C}(x, n_i) \mid x \rightarrow n_i\}$
8 $\quad d'(n_i) = \max(d(n_i), dpred)$
9 **return** T'

Smooth recovery policy

The smooth recovery policy tries to lead the system back to a reference timetable, and uses time margins to compensate deviation, as in schedule policy. However, instead of trying to return to the reference timetable as soon as possible, this policy tries to smoothen the deviation recovery process. When a train deviates from its planned schedule, the delay creates a time gap, i.e. a change in the normal distance and time headway to its predecessors and successors. To reduce this gap, a subset of the preceding trains and of the successor trains is selected. The speed and dwell times of these selected trains are adapted to fill the gap that the delayed train has created. Furthermore, in order to avoid creation of time and space gaps around selected trains, the changes applied to their dwell or running times are weighted: trains that are close to the initial delayed train are subject to important changes, and farther trains are less affected. These weights are determined empirically.

Local adjustement policy

This policy is another elaborate traffic management policy that consists in compensating deviations by trying to get as close as possible to the reference timetable while ensuring reasonable headways. Following a deviation on a train, a hold signal is sent to the trailing or leading train—or both—w.r.t. the affected train. These trains will then themselves send hold signals to their trailing or leading trains. In doing so some signals may cancel each other or reduce the holding time at station level. This is an example of a policy where the combination of local actions can lead to a global improvement of the quality of service. This is a rule-based policy in which the amount of holding time is determined by

comparing the deviations with a set of thresholds. As for the smooth recovery policy, this policy is tuned empirically.

3.4 Performance evaluation

To assess the performance of traffic management techniques and, more generally, to measure the quality of the provided service, operators compute Key Performance Indicators (KPIs for short) from recorded logs. KPIs give a quantitative measure of punctuality of services, of their regularity, of passengers waiting times, etc. Furthermore, KPIs are also used to establish contracts between stakeholders of public transports and operators. In many cities, contracts fix thresholds for some KPIs. Failing to meet the recommended thresholds the result in financial penalties.

The metro division of the international organization for public transport (UITP) has defined a set of KPIs to measure the quality of the provided service in URSs [5]. These KPIs aim to guarantee service quality with objective indicators computed from logs recorded during operation of the network (e.g., number of trips that are delayed by more than x minutes), but also consider service quality perceived by passengers (e.g., estimated waiting times).

Examples of KPIs are the following.

3.4.1 Service punctuality

Punctuality of service in transport systems is one of the most important criteria. The UITP defines punctuality indicators depending on the frequency of the considered service. It divides urban rail service into two categories: high-frequency and low-frequency services.

High-frequency services. When a URS is operated at a high frequency (e.g., one train leaving each station every 2 minutes [1]), measuring punctuality as the adherence of trains departure or arrival dates to a fixed reference timetable is irrelevant. As high-frequency services are usually provided at peak hours, what really matters to passengers in this situation is the absence of fluctuations in trip durations.

Let us suppose that the set of all trips (from a departure terminus to an arrival terminus) are indexed with $i = 1, 2, \ldots, n$, where n is the total number of trips. Let us also denote by δ_i the total duration of trip i in the reference timetable, and $\dot{\delta}_i$ the realized total duration of trip i. The high-frequency service punctuality criterion is then computed w.r.t. a given tolerance threshold ε (expressed in seconds), and is given by

$$\textsc{Kph}(\mathscr{T}, \dot{\mathscr{T}}) \triangleq \frac{1}{n} \cdot \left| \bigcup_{i=1}^{n} \{\dot{\delta}_i \mid \dot{\delta}_i - \delta_i \leq \varepsilon\} \right|.$$

Intuitively, it is the ration of trips that are performed within their reference duration, up to a tolerance of ε time units.

[1] No rigorous definition of low and high frequency is given by the UITP.

Low-frequency services. In low frequency services (for instance when one train leaves a station every 20 minutes), passengers rely on published schedules and want trains to arrive and leave at their scheduled times. The corresponding KPI measures the proportion of train arrivals that occur on time (w.r.t. the reference timetable), i.e. with a delay smaller than a threshold ε. This KPI can be defined for all stations or just for a subset of important nodes in the network. Let us suppose that all arrivals at the considered stations are simply indexed by $i = 1, 2, \ldots, n$, with n the number of all considered arrivals. Denoting d_i^a the date of the i^{th} arrival in the reference timetable $\mathscr{T}$, and $\dot{d}_i^a$ the date of the corresponding effective arrival in the realized timetable $\dot{\mathscr{T}}$, the formula for this KPI is

$$\textsc{Kpl}(\mathscr{T}, \dot{\mathscr{T}}) \triangleq \frac{1}{n} \cdot \left| \bigcup_{i=1}^{n} \{\dot{d}_i^a \mid \dot{d}_i^a - d_i^a \leq \varepsilon\} \right|.$$

3.4.2 Service regularity

Regularity of service can be measured from two different perspectives: it can be considered in terms of *passenger waiting times* at stations, or in terms of *headways*.

Waiting times. This measure reflects passengers' experience and is a frequently used indicator according to the UITP [5]. It estimates the proportion of passengers that have to wait for a train more than x minutes at a subset of specified stations. It is measured based on planned headways and thresholds on acceptable waiting times.

Let us suppose that the considered stations are indexed by $k = 1, 2, \ldots, m$, and that departures from each station k are indexed by $i_k = 1, 2, \ldots, n_k$. Let $d_{i_k}^d$ denote the reference date of the i^{th} departure from station k. We define the i^{th} reference arrival headway by $h_{i_k}^d \triangleq d_{i_k+1}^d - d_{i_k}^d$, i.e., the difference between two successive arrival dates at station k in the reference timetable $\mathscr{T}$. Similarly, $\dot{h}_{i_k}^d \triangleq \dot{d}_{i_k+1}^d - \dot{d}_{i_k}^d$ defines the realized headway. The passenger waiting times KPI is then defined by the formula

$$\textsc{Krw}(\mathscr{T}) \triangleq \sum_{k=1}^{m} \sum_{i_k=1}^{n_k} \frac{wf_{i_k} \cdot wt_{i_k}}{\dot{h}_{i_k}^d},$$

where

- wt_{i_k} is the (maximum) acceptable waiting time for a headway equal to $h_{i_k}^d$, and
- wf_{i_k} is a weighting factor that reflects passenger demand. It is function of the reference headway, and varies according to the time of the day and week. It is required that $\sum_{k=1}^{m} \sum_{i_k=1}^{n_k} wf_{i_k} = 1$.

Headway. Regularity of service can also be measured as the ration of realized headways that are the durations planned in the timetable. This KPI calculates the percentage of durations between departures that are close to the reference headway $h_{i_k}^d$, with a tolerance of $\pm\varepsilon$. Denoting $\dot{d}_{i_k}^d$ the date of the i^{th} effective departure from station k, and

$\dot{h}^d_{i_k} = \dot{d}^d_{i_k+1} - \dot{d}^d_{i_k}$ the i^{th} effective departure headway, the headway KPI is given by

$$\textsc{Krh}(\mathscr{T}, \dot{\mathscr{T}}) \triangleq \frac{1}{n} \cdot \left| \bigcup_{k=1}^{m} \bigcup_{i_k=1}^{n_k} \left\{ \dot{h}^d_{i_k} \mid \dot{h}^d_{i_k} \in [h^d_{i_k} - \varepsilon, h^d_{i_k} + \varepsilon] \right\} \right|.$$

3.4.3 Other indicators

Punctuality and regularity are common criteria to assess service quality in transport systems. The UITP proposes several other criteria, we list some of them below:

Service availability. This KPI measures how a service provider's complies with the promised transport supply. It is the ratio of car-kilometers produced divided by the number of car-kilometers planned. The number of car-kilometers measures the total distance run by trains, and is computed as the number of trains multiplied by the average length of their trips in kilometers.

Service reliability. This KPI computes the distance (in kilometers) ran by trains between important incidents, i.e between two train cancellations or between incidents causing delays of more than x minutes. It is the ratio:

$$\frac{\text{number of car-kilometers produced}}{\text{number of failures leading to a delay} > x}.$$

Passenger density. This KPI measures the percentage of passengers that benefit from a sufficient space during their trips. Imposing a passenger density of x requires that trains are loaded with less than x passengers per square meter. The value of x reflects the worst acceptable crowding conditions. This parameter usually ranges from 3.5 to 5.5 pgr/m^2. Note that this KPI is a subjective one, as the acceptable value for x depends on the considered city.

Remark 3.3
From the above examples, one can see that KPIs are parameterized : they are defined according to thresholds, that are negotiated among stakeholders of metro management. Some KPI Measures are used only for passenger information, for statistics. However, in some cases, these KPIs are contractualized and failing to provide the promised quality results in financial penalties. Of course, UTS operators are not liable to delays originating from *force majeure* events such as passenger sickness, natural disasters,... The KPIs presented above are the ones recommended by the UITP [5]. However, many measures can be invented and evaluated from logs of operation days, so operators and stakeholders can decide to use other KPIs. Operators also have their own internal indicators for performance evaluation. ◊

3.5 Block signaling systems

Trains in motion are large, heavy and fast objects. The coefficient of friction between wheels and rails is low. As a result, if a danger of collision is detected, a train cannot

be stopped or rerouted quickly to avoid an accident. For this reason, it is of paramount importance to carefully build and update trains trajectories in such a way that trains are separated by a distance allowing emergency braking. This distance is called the *safety headway* and is usually expressed in time. A safety headway is a duration that is larger that the time needed by a train to get immobilized after it starts braking. It depends on several parameters such as speeds of trains, their lengths, their braking capacities, reaction times, etc. In order to ensure that safety headways are respected, URSs rely on *block signaling systems*, i.e., mechanisms that define *blocks* and impose that each block be occupied by at most one train at the same time. Block signaling systems fit into two categories: FB systems and MB systems. We hereafter provide a presentation of both systems.

3.5.1 Fixed-block systems

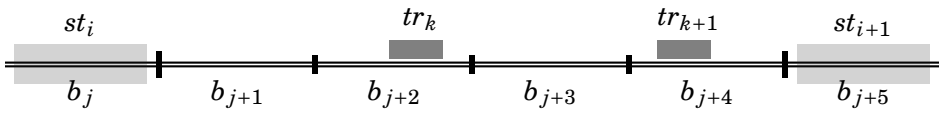


Figure 3.3: Fixed-block policy principle

In FB signaling systems, the blocks are portions of tracks usually delimited by signals. A block is considered occupied as soon as a part of a train is located in this track portion. In old rail systems, trains' localization is performed with track circuits that only indicate if a train is present on a block, without giving its exact position. As a consequence, a train can occupy two consecutive blocks at the same time. In high speed URS, safety imposes to leave a free block between two consecutive trains running at maximal speed. The example shown in Figure 3.3 shows two trains traveling in the same direction and separated by a block. In these kind of systems, 3-aspect signals are used instead of green/red 2-aspect signals. Green signal means *proceed normally* and ensures that the signal after the block which the train is entering is not red; yellow (or orange) signal means *proceed with a reduced speed* as next signal is red; and red means *stop the train*.

Block sizes vary depending on the frequency of the service and are carefully calculated to optimize the traffic flow. Among the parameters that have to be taken into account when calculating the lengths of blocks, we can find: the length of trains, the maximum possible speed over the line sections, different trains' speeds, braking characteristics of trains, sighting distances, and an estimation of reaction times.

FB signaling systems is a reliable technique for ensuring safe trains' spacing. However, it is a pessimistic approach that operates systems at capacities lower than what they can actually handle.

3.5.2 Moving-block systems

In recent URS systems, moving blocks signaling systems are replacing fixed block systems. The moving block policy as described by Pearson [55] states that "*A train is con-*

tinuously supplied with accurate information of the position of the nearest obstacle on the track ahead of it [...] it may be a preceding train, which itself may be moving or stationary. The speed of the train is constantly checked and adjusted [...] so that it is always possible for the train to be brought to rest without colliding with the obstacle." Moving blocks are defined as virtual boxes that envelop trains and move with them. The size of the envelope can also be dynamically adapted to the current speed of a trains. Safety is ensured by adjusting trains speeds in such a way that envelopes never overlap, even in case of emergency braking. Figure 3.4 gives an example of moving block: two trains tr_k and tr_{k+1} are moving in the same direction. The dotted boxes around trains are their envelopes.

For a MB approach to be applicable, trains' exact positions must be known in real-time. For that, operators rely on communication-based train control (CBTC) systems: automatic train control systems based on the continuous communication between trains through radio signals.

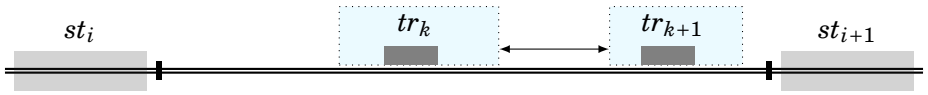


Figure 3.4: Moving-block policy principle

Moving block systems allow for shorter headways than fixed block systems. In particular, the headway between two trains is usually shorter than the time needed by a train to cover a whole physical block. To compare FB and MB systems, HILL and BOND used a discrete event simulator written with the SIMSCRIPT simulation language [37]. They simulated 3 trains running along a straight line with four stations and a fixed dwell time with fixed block ans moving block signaling, and computed the minimal achievable headways in function of maximum allowed speeds of trains. An achievable minimum headway is the minimum possible headway for which trains' movements are not disturbed due to spacing constraints. They showed that these minimum headways are smaller in the system with MB signaling than with FB signaling.

3.6 Glossary

To facilitate further reading, we provide a list of definitions for the terms frequently used in document related to Urban train systems.

Block A physical or virtual portion of tracks that must be occupied by at most one train at all instants.

Carousel A circuit forming a loop that trains run through, shuttling back and forth. It can, for instance, be a loop among others of a single commercial line.

Depot A large train parking area.

Deviation A delay or an advance w.r.t. a planned schedule.

Dwell time The time a train spends on a station's platform for the alighting and boarding of passengers. It includes the time needed to open and close the doors.

Fork A Y-shaped track configuration at which two (or more) train paths diverge.

Headway The time difference between the passage dates at a given point of two consecutive trains traveling in the same direction.

Headway (safety) The minimum headway that ensures that no head-to-tail collision between trains occurs, and avoids the recourse to unnecessary emergency braking.

Junction A Y-shaped track configuration at which train paths converge. It can be seen as the opposite of a fork.

Key performance indicator A performance measurement formula.

Operation We use the term *operation* to describe either trains dwelling at stations or running between stations. We, therefore, have two types of operations: dwell operations and running operations.

Maneuver We call *maneuver* any series of movements that define trains turnbacks at termini, trains insertion or extraction from a line, etc.

Running time The time between the moment a train starts leaving a station and the moment it completely stops at the next one.

Timetable A description of an order on train operations (departures and arrivals), and of their occurrence dates.

Timetable (active) A timetable used during operation for traffic management. It can evolve during the day.

Timetable (realized) A timetable in which all events have already occured.

Timetable (reference) A timetable describing the ideal behavior of the system.

Time margin A duration that can be subtracted from a dwell or running time to compensate a deviation.

Trip A series of operations that start with a departure from a given terminus and end with an arrival at a destination terminus.

Trip chaining Linking two successive trips, usually by a turnback maneuver.

Route A virtual path of trains passing by a succession of stations.

Service A set of chained trips. Each train has to be associated with a service to perform trips. At the beginning of an operation day, trains move from a depot to the commercial lines to begin their services; and at the end of their services, they are either brought back to the depot or parked at parking platforms to be reinserted later.

Terminus A station from which a trip starts or at which a trip ends.

4

State of the art

This section is a state of the art. It focuses mainly on models used in the litterature to represent compute and maintain timetables, on models used to represent objects routing in a network (mainly Petri net variants) and last on queuing networks, used as a way to compute performance of railway systems.

4.1 Timetables: modeling and design

Timetables are a simplified models of train movements between stations during a fixed period (usually a few hours or an operation day). They are central components for traffic control in rapid transit systems (RTS). They describe train trajectories, specify the desired departure and arrival dates of trains and also fix an order on trains passage at forks and joins and in shared sections.When a day, week or month of operation is complete, timetables are used as references for the analysis of achieved performance.

Timetables also play a role in safety, and to guarantee performances: as already seen in chapter 3 trains are subject to safety constraints, to service quality constraints, etc. A way to fulfill these constraints is to specify train trajectories in advance in a reference timetable that represents the ideal behavior of the system, and to operate the network is such a way that it realizes this timetable. In practice, operators define a set of timetables for each line. Each timetable is used in a specific context: working days, weekends, holidays, etc. For example, train departure headways in timetables are generally longer during weekends than during working days. The construction of these timetables is a nontrivial task that amounts to solving an optimization problem called *the timetabling problem*.

Formally, a timetable is a set of events, with an ordering, and a function that associates a date to each of these events.

Definition 4.1. A *timetable* (TT) is a tuple $TH = \langle \mathscr{G}(S), \Delta_0 \rangle$ where $\mathscr{G}(S) = N, \longrightarrow$ is an acyclic graph and $\Delta_0 : N \rightarrow \mathbb{Q}_{\geq 0}$ is a function that associates a desired (arrival or depar-

ture) date to each node of the solution. We furthermore require that when there is an arc from n_i to n_j, $\Delta_0(n_j) - \Delta_0(n_i) \geq \lambda_{ij}$.

The requirement on function Δ_0 indicates that the timing constraints must be preserved. Note that Δ_0 is not unique. A possible function is the dating function that attaches to each node the earliest execution date to each node n, i.e. the maximal sum of weights along a path from the initial node to n. However, it is obvious that this solution is not robust : any delayed event n_i impers the execution dates of the next arrivals and departures, that must occur at earliest at the date of n_i plus the time constraint imposed by edges leaving n_i. A preferable solution is to consider a buffer time b, and to attach inductively dates to nodes : a node n_i can be attached a date $\Delta_0(n_i) = \max_{e_{ji}}(\Delta_0(n_j) + \lambda_{ji}) + b$, that is the earliest date allowed by the predecessors of n_i, plus a time buffer b, that allows the represented event to occur late without delaying its successors. This way, the time buffer b may avoid constant re-evaluation of execution dates of events after each delay.

4.1.1 The timetabling problem

Given a desired rail service expressed as a set of train trips subject to constraints such as: trains' speed profiles, dwell times at stations, design headways, drivers' schedules, energy consumption, etc., the timetabling problem consists in finding an optimal schedule for train departures and arrivals that meets all constraints. The value to optimize can be for instance a KPI.

The timetabling problem can be expressed as a mixed integer linear programming (MILP) problem. The canonical formulation of MILP problems is the following:

$$\begin{array}{rl} \min_{x,y} & c^T x + h^T y \\ \text{subject to:} & Ax + Gy \leq b \\ & x \geq 0 \\ & y \geq 0 \text{ (integral)} \end{array}$$

A and G are matrices; x, y, c, h and b are vectors; $c^T x + h^T y$ is the objective function to minimize; x contains variables that can take non-integer values, and y variables that must take integer values.

A natural question once the timetabling problem has been defined is: why not reschedule departures and arrivals online with MILP to achieve the best possible KPI at the end of an operation day ? Though this idea is theoretically relevant, it cannot be applied in practice. It is well known that MILP (and hence the timetabling problem) is NP-complete (see for instance [64]). Further, for a standard metro line with 40 trains and 20 stations running with the highest possible performance, the number of departures and arrivals rapidly (and hence the number of variables) rapidly exceeds a million. A requirement of regulation is to **provide fast answers**: the time that can elapse between a train arrival in a station and the decision of a regulation algorithm should not exceed the minimal dwell time at this station, and is usually bounded to a few seconds. Timetabling are particular kinds of MILP problems, with simple linear constraints (of the form $d_{i+1} \geq d_i + c$, where c is a constant), where one can expect efficient resolution techniques. Nevertheless, due to the size of the problems to solve, MILP cannot be used in real-time for regulation. Further, searching for optimal dates for a period that exceeds 30 minute is very often useless, and delays are very frequent, and hence force to reconsider scheduled dates at

almost every arrival. Hence, an optimal solution computed at date t has few chances to be consistent 30 minutes later.

As a consequence, the preferred use of timetabling solutions is to find optimal (or quasi optimal if the problem to solve is too big) timetables, and then rely of regulation algorithms to stick to this reference schedule. In the rest of this section, we list several models that have been used to represent and maintain timetables, and techniques to compute optimal or quasi-optimal timetables.

4.1.2 Models

Space-time graphs

A simple and common representation of timetables is the *space-time graph*. It is a two-dimensional diagram in which abscissae represent locations (stations) in the rail network, and ordinates represent time. On this diagram, train trajectories are drawn as paths between stations: a point in the diagram is always part of a given train trajectory and depicts its presence at a station at a certain point in time; an edge connecting two points $p_0 = (x_0, y_0)$ and $p_1 = (x_1, y_1)$ describes the timed movement of a train between the two stations corresponding to p_0 and p_1. Note that if $x_0 = x_1$, then the train is dwelling and stays at the associated station for a duration $y_1 - y_0$. Formally, a trajectory T_k for a train k is a sequence of points $T_k = p_0.p_1.\ldots p_m$.

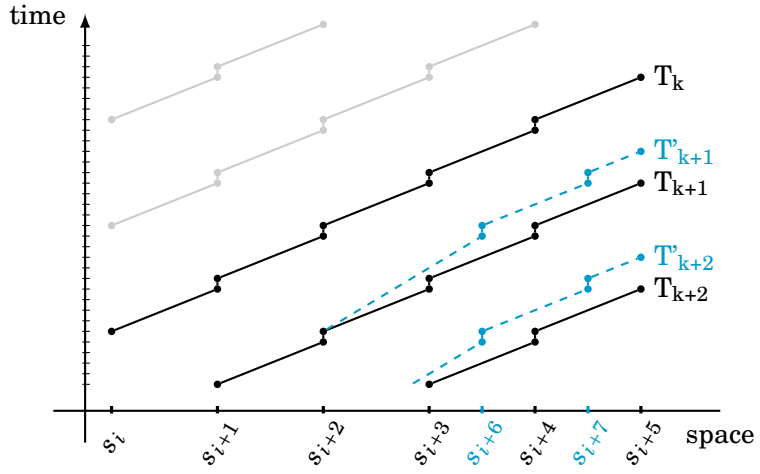


Figure 4.1: Example of a space-time graph with alternative paths

CACCHIANI et al. [12] rely on this representation to show the ideal timetable and the real (effective) timetable for a set of trains. They describe a rail network by a graph $N = \langle S, E \cup A \rangle$ where $S = \{s_0, s_1, \ldots, s_{m-1}\}$ is a set of nodes, $E = \{e_0, e_1, \ldots, e_{n-1}\}$ a set of edges, and $A = \{a_0, a_1, \ldots, a_{p-1}\}$ a set of arcs. Each node s_i represents a station; an edge e_j is a bidirectional track of the form $e_j = (s_a, s_b)$ with $(a, b) \in \{0, 1, \ldots, m-1\}^2$. An arc is a track that can be traveled in only one direction. A *train trajectory* is a directed path of N, with dates associated to arrival and departures in stations. A timetable is defined as the union of trajectories for a set of considered trains. It is also possible to represent alternative timetables for a train, that can be used if the optimal initial one cannot be realized.

Figure 4.2 shows a partial timetable with highlighted trajectories of 3 trains indexed by k, $k+1$ and $k+2$. Ideal trajectories are depicted in solid lines and alternative ones in dashed lines. Normally, to complete the timetable, two fictitious nodes are added to symbolically represent the origin and destination of trains. In this scenario, trains tr_{k+2} and tr_{k+1} will ideally pass by stations s_{i+3} and s_{i+4} to reach s_{i+5}, but also have the possibility of taking the alternative path $s_{i+6} \rightarrow s_{i+7} \rightarrow s_{i+5}$.

Stepped blocking-times diagrams

In 1959, HAPPEL [33] introduced a method to represent occupation times of blocks by trains in a fixed-block system. Using this method, each train trajectory is represented by a blocking-time *stairway* in a space-time diagram, as illustrated in Figure 4.2. Steps of the stairway include all safety margins needed to ensure that no collision occurs. For this condition to hold, all stairways must be disjoint in the space-time diagram. Safety margins include: route formation time (time between train route request transmission and route reply reception), sighting time, approaching time to the next signal, clearing time (time between the moment when the train's head leaves a block and when its tail leaves the same block), and route release time. These steps can be seen as the occupation of the network by trains, or in a more quantitative way as a consumption of line *capacity* by trains. Here capacity is the ability to move a specific amount of traffic over a given route within a timeframe.

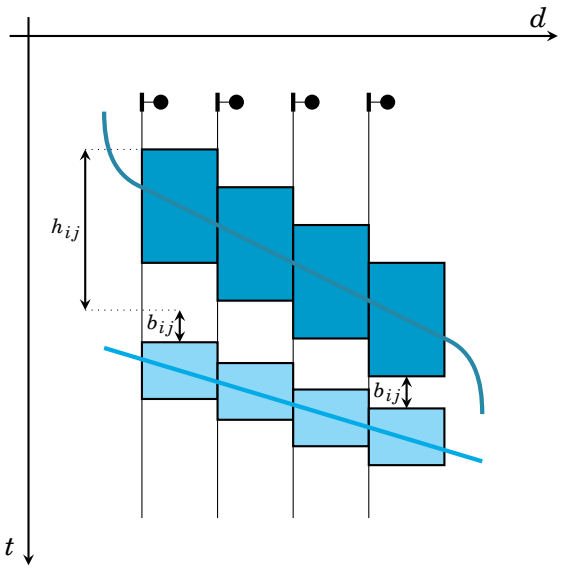


Figure 4.2: A stepped blocking-times diagram

This model has later been used to compute timetables with ADLER's concatenation method [2]. The main idea is to push "stairways" as close as possible to each other while making sure there is no overlapping between stairs. This technique compute timetables with the minimal capacity consumption for a set of train movements. More details on this technique can be found in the article on railway infrastructure capacity assessing methods by BÜKER [11].

Figure 4.2, shows blocking-times for two successive trains tr_i and tr_j. h_{ij} represents the imposed safety headway between the two trains while b_{ij} represents a time buffer used avoid propagation of delays from tr_i to tr_j. In fact, concatenation of blocking-times with-

out considering buffers, produces a timetable in which schedules of all successor trains of a delayed metro must be recomputed when an incident occurs, regardless of the amplitude of the delay. So, this type of optimal timetable is not *robust* enough to delays to be operational. One can easily see that inserting buffers is important, and that robustness of a timetable can decrease when its capacity improves. Introduction of buffers increases the number of variables and hence the complexity of the timetabling problem.

Alternative graphs

The model of *alternative graphs* was originally proposed by MASCIS and PACCIARELLI [47, 48] as a means to design and solve jobshop scheduling problems (JSP). In this type of problems, a finite set of *jobs* (tasks) have to be assigned to a set of *resources* (machines). The objective is to minimize the makespan, i.e., the time needed to process all jobs.

Rail scheduling problems can be assimilated to jobshop problems where jobs are trains movements and resources are track portions. This model was used by D'ARIANO et al. [18, 19] to address the problem of real-time optimization of rail traffic in mainlines after disturbances. An alternative graph is a compact representation of *all* possible schedulings of tasks in system that satisfy a set of predetermined constraints.

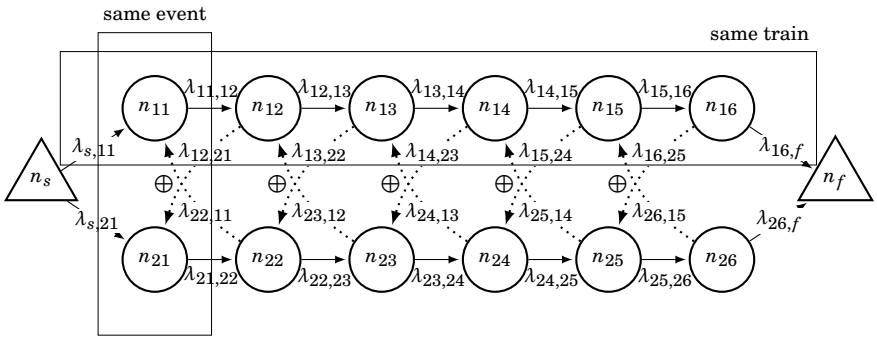


Figure 4.3: Example of an alternative graph

Formally speaking, an alternative graph is a tuple $\mathcal{G} = \langle N, F, A, \Lambda \rangle$ where

- N is a set of *nodes*. Each node is graphically represented by a circle and models a particular *operation*. Operations last for given durations that can, for instance, be the running time of a train on a rail track portion or its dwelling time at a station. In addition to these nodes, two dummy nodes, n_s and n_f, are included in N. They respectively indicate the beginning and the end of the modeled schedule.
- F is a set of *fixed arcs*. An arc is a node $(n_i, n_j) \in N^2$ that indicates the existence of a time constraint between the beginning of the operation (modeled by) n_i and the beginning of operation n_j. Fixed arcs represent constraints that are always valid and have to be taken into account in all schedules alternatives. (e.g., the minimal running time of a train to cover a track portion.) Graphically, fixed arcs are drawn as solid arrows.
- A is a set of **pairs** of *alternative arcs*. Alternative arcs indicate possible priorities between pairs of events. In the context of rail systems, these events are of different trains. These arcs come by pairs and the constraints they model are mutually exclusive. We graphically represent alternative arcs by dotted arrows with an exlusive-or (xor) symbol near each pair. Formally a pair of alternative arcs in A is a pair (e_{ij}, e_{kl}) where $e = (n_i, n_j)$ and $e_{kl} = (n_k, n_l)$ are pairs or nodes.

We will denote by E the set of all arcs of the alternative graph; namely, fixed arcs of F and alternative arcs in the pairs of A. $E = F \cup \{e \mid \exists (e,e') \in A \vee \exists (e',e) \in A\}$. For simplicity, we will use the notation e_{ij} to designate an arc from n_i to n_j, instead of (n_i,n_j).

- $\Lambda : E \to \mathbb{Q}_{>0}$ is a function that associates a strictly positive rational value to each arc. This value defines a time constraint on the occurrence dates of the events at the origin and destination of an arc. Again, we shall write λ_{ij} to designate the time constraint $\Lambda(e_{ij})$ between nodes n_i and n_j. This constraint represents the duration of the operation corresponding to node n_i; it also represents the minimum time that has to elapse before operation n_j can start.

In a metro network, track portions delimited by blocks, forks and joins can be considered as resources, that can be used by at most one train. Some operations in an alternative graph use the same resources, and hence cannot occur concurrently. One can think for instance of two operations represented by nodes n_i, n_j that depict respectively train t_i entering a fork section and train t_j entering the same fork. Clearly, n_i has to occur before n_j or n_j before n_i to avoid a collision. The situation where two or more trains ask to enter the same track portion is called *a conflict*. When a conflict exists, one can represent the need for a scheduling of operation with an alternative arc $\big((n_i,n_j),(n_j,n_i)\big)$. Solving one conflict means choosing one arc or the other in the alternative. Generalized to the whole set of conflicts in the alternative graph, this problem is called the conflict resolution problem (CRP).

Formally: given an alternative graph $\mathcal{G} = \langle N, F, A, \Lambda\rangle$, a *selection* of alternative arcs is a set of arcs obtained by choosing only one arc from each pair of arcs in A. A selection is *complete* if it contains exactly one arc from each pair. It is a solution if the graph $\mathcal{G}(S) = \langle N, F \cup S, \Lambda\rangle$ does not contain cycles (cycles denote contradictory constraints of the form *"a occurs before b and b occurs before a"*). Conflict resolution consists then in finding a complete and consistent selection for $\mathcal{G}$. Once a solution is found for an alternative graph, building a consistent schedule consists in choosing a date d_n for each node n such that the dates meet all constraints. If one wants, for instance, the earliest completion dates for arrival and departures of trains, an easy adaptation of the flooding algorithm 3.1 depicted in Chapter 2 can associate a date to each node in a time that is linear in the size of $F \cup S$.

D'ARIANO et al. propose a CRP resolution approach based on a branch and bound algorithm coupled with a set of heuristics to improve computation speed [18] to build timetables. The algorithm returns a conflict-free scheduling of trains and at the same time minimizes a criterion such as the maximal realization date of all events. Notice, however, that if the criterion chosen is minimization of completion date for a timetable, then dates chosen for nodes are the earliest possible ones, and the timetable obtained is not robust to even small perturbations.

Systems of inequalities

Alternative graphs can be modeled equivalently as a system of inequalities in which variables can either be real valued variables $x_1, \ldots, x_n$ where x_n is a variable for node $n \in N$, or boolean variables $b_1, \ldots, b_m$ where b_a is a variable for pair of arcs $a \in A$ (conflicting pairs of pairs of nodes). The system then consists in inequalities of the form $x_{n'} - x_n \geq \Lambda\big((n,n')\big)$ for fixed arcs, statements of the form $b_a \in \{\mathsf{true}, \mathsf{false}\}$, and constraints of the form $(b_a = \mathsf{true}) \Rightarrow \big(x_{n_{11}} - x_{n_{00}} \geq \Lambda\big((n_{11}, n_{00})\big)\big)$ and $(b_a = \mathsf{false}) \Rightarrow \big(x_{n_{01}} - x_{n_{10}} \geq \Lambda\big((n_{01}, n_{10})\big)\big)$, denoting

the fact that some constraints on dates depend on the order of train passages in shared sections. A timetable is then an real valuation for x variables and a boolean valuation for b variables. Finding a valuation for variables in this system obviously produces a solution for an alternative graph, and gives an occurrence date for every node.

4.2 Formal models for objects routing

Tests performed directly on real systems, when feasible, tend to be extremely costly and time consuming. Systems are, thus, generally abstracted to a model, and with a certain degree of simplification. Formal modeling of systems allows for automated analysis of their behaviors. Models for rail systems share many aspects with manufacturing systems and production lines. Many models for discrete event systems with resources and time exist. In this section, we focus on Petri nets variants. The reason is that Petri net naturally model independence among actions, which is natural in railway operations representation, consider resources, and all build on a natural graphical representation in which objects (tokens, colored tokens, ...) are moved from a place to another. This family of formalisms seems well adapted to urban train systems. In the following, we present several variants of Petri nets, with time and randomness, and explain why these existing models cannot be used directly to model metro networks.

4.2.1 Petri nets

Petri nets are a powerful modeling formalism that has long been used to model, analyze and control DESs. It is particularly adapted to model systems with concurrent and asynchronous processes. Petri nets are a good compromise between expressiveness and decidability: they are more expressive than finite automata, but yet, several properties such as, termination, boundedness, coverability [40] (see also [28]) and even reachability [49] are decidable properties. This allows, in particular, to verify that an undesirable state of a system is not accessible. Similarly, Petri nets allow for control techniques [27] to ensure that the modeled system never reaches a bad state. Petri nets can model resources (modeled as tokens in places) mutual exclusion in concurrent systems,... Last, they can be represented as graphs, which aspect is often close to the system they model.

The origin of Petri nets dates back to C. A. PETRI's Ph.D. thesis [57]. There exist numerous variants of Petri net models. Therefore, when simply speaking about *Petri nets*, we generally refer to the model defined in this section that we will call the *classical* model.

A net is a bipartite directed graph and is defined as follows:

Definition 4.1 (net)
A *net* is the tuple $\langle P, T, A\rangle$ where $P = \{p_0, p_1, \ldots, p_{n-1}\}$ is a nonempty set of *places*, $T = \{t_0, t_1, \ldots, t_{m-1}\}$ is a nonempty set of *transitions*, and $A \subseteq (P \times T) \cup (T \times P)$ is a set of *arcs* (also called the flow relation). ◊

We also assume that places and transitions are distinct sets, i.e., $P \cap T \triangleq \varnothing$, and that each place in a net (resp. transition) is connected to a transition (resp. a place), i.e., $\forall x \in P, \exists y \in T, \langle x, y\rangle \in A \vee \langle y, x\rangle \in A$, and $\forall x \in T, \exists y \in P, \langle x, y\rangle \in A \vee \langle y, x\rangle \in A$.

Places in a net model resources and states of components of the modeled system. Brought back to urban train systems, a place can model a track portion in a rail system. Transitions represent events such as departures or arrivals of trains.

Definition 4.2 (Petri net)
A Petri net is a tuple $\langle P, T, A, W, M_0 \rangle$ where:

- $\langle P, T, A \rangle$ is a net;
- $W : A \to \mathbb{N}_{>0}$ an arc *multiplicy* (or arc weighting) function; and
- $M_0 : P \to \mathbb{N}$ is the *initial marking*. ◊

In a Petri net, a *marking* assigns a natural number of *tokens* to each place. We denote by $M(p)$ the number of tokens that marking M associates to place p. Intuitively, a token in a place can be seen, for example, as a train in a station or a piece in a machine. In doing so, the marking gives a representation of local states of the underlying system, and its global state.

Petri nets are a dynamic model in which the *initial marking* represents the initial global state of the modeled system. From this state, it is possible to perform actions that lead to a change of marking. Actions are performed by *firing* transitions. This operation consumes tokens from places and produces new tokens in other places. Arcs in A dictate the flow of tokens, and W indicates the number of tokens to be consumed and produced during transition firing.

Definition 4.3 (preset and postset)
We define the *preset* (preconditions set) of a given transition t as the set of input places of t: ${}^\bullet t \triangleq \{p \mid \langle p, t \rangle \in A\}$. Similarly, the *postset* (postconditions set) of a transition t is the set of its output places: $t^\bullet \triangleq \{p \mid \langle t, p \rangle \in A\}$.

This notation can be used for places: for each place $p \in P$, ${}^\bullet p$ denotes the set of transitions that may produce tokens in p, and $p^\bullet$ denotes the set of transitions that may consume tokens from p. ◊

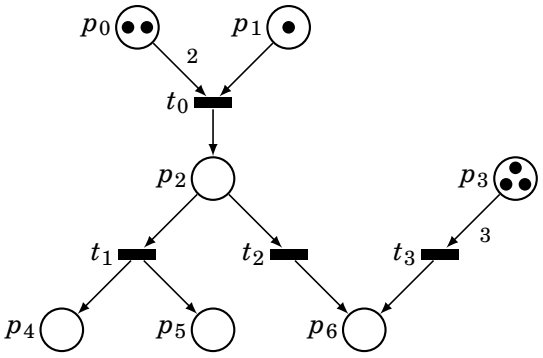


Figure 4.4: Example of a Petri net

Figure 4.4 is a graphical representation of a Petri net. Places are represented by circles, transitions by filled rectangles, arcs by arrows, tokens by dots, and arc multiplicity by numbers on top of arcs (usually omitted when equal to 1). Places and transitions may be labeled to indicate which conditions and actions they represent in the underlying system. Markings are represented by putting a certain number of tokens in the marked places. On this example, $M(p_0) = 2$ which is symbolized by two tokens in place p_0.

Remark 4.1 (basic Petri net). In the particular case where all arcs have weight 1, W is usually dropped from definitions and is implicit from the flow relation. In this case, we will say that the considered Petri net is *basic*.

Semantics

Petri net semantics define *moves* between markings. Moving from a marking M to a new marking M' is performed by *firing* a transition $t \in T$. Firing a transition models the execution of the event corresponding to t in the underlying system. A transition t can fire only when it is *enabled*.

Definition 4.4 (enabledness)
A transition $t \in T$ is enabled by a marking M iff $\forall p \in {}^\bullet t, M(p) \geq W(p,t)$. The set of all transitions enabled by a marking M is denoted $\mathsf{enab}(M)$. ◊

Definition 4.5 (firing)
Firing an enabled transition t from a marking M consists in:

1. consuming $W(p,t)$ tokens from each place p in its preset ${}^\bullet t$, leading to a temporary marking M_{tmp},

$$\forall p \in P : M_{\text{tmp}}(p) \triangleq \begin{cases} M(p) - W(p,t) & \text{if } p \in {}^\bullet t, \\ M(p) & \text{otherwise} \end{cases}$$

2. producing $W(t,p)$ tokens in each place p in its postset $t^\bullet$, leading to a new marking M'.

$$\forall p \in P : M'(p) \triangleq \begin{cases} M_{\text{tmp}}(p) + W(t,p) & \text{if } p \in t^\bullet, \\ M_{\text{tmp}}(p) & \text{otherwise} \end{cases}$$

We will writes $M \xrightarrow{t} M'$ when firing t from marking M produces marking M'. ◊

From this firing rule, we introduce the two convenient notions of *persistency* and *new enabledness*.

Definition 4.6 (persistency and new enabledness)
Given a marking M and an enabled transition $t \in \mathsf{enab}(M)$, we define the set of persistent transitions, after firing t from M, as:

$$\mathsf{pers}(M,t) \triangleq \mathsf{enab}(M_{\text{tmp}}) \setminus \{t\}.$$

and the set of newly enabled transitions as:

$$\mathsf{newl}(M,t) \triangleq \mathsf{enab}(M') \setminus \mathsf{pers}(M,t).$$ ◊

Remark 4.2 (safe and elementary net). When there exists a bound K such that, for every marking M of a basic Petri that is reachable from the initial marking M_0, and for every place $p \in P$, $M(p) \leq K$, we say that the net is *K-bounded*. When a net is 1-bounded, it is called a *safe* Petri net. Last, a semantic variant of Petri nets allows firing of a transition t only if, for every place $p \in t^\bullet$, $M(p) = 0$ (one cannot produce a resource in an already occupied place). Nets with this semantic variant are called an *elementary* net.

Petri nets with inhibitor arcs

This model adds *inhibitor arcs* [56]. These arcs add the possibility to condition firing of a transition to absence of tokens in some places.

Definition 4.7 (Petri net with inhibitor arcs)
A Petri net (with inhibitor arcs) is a tuple $\langle P, T, A, {}^\circ A, W, M_0 \rangle$ where P, T, A, W and M_0 have the same meaning as in Petri nets, and ${}^\circ A \subseteq P \times T$ is a set of inhibitor arcs. We will assume that inhibitor and flow arcs are exclusive, that is an arc (p,t) is either in ${}^\circ A$ or in A. Given a transition $t \in T$, we denote by ${}^\circ t$ the set of *inhibiting places* of t. ${}^\circ t \triangleq \{p \mid \langle p,t \rangle \in {}^\circ A\}$. ◊

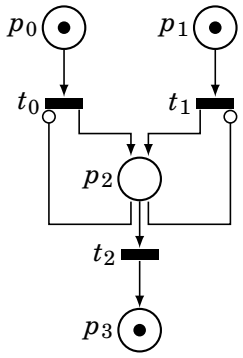


Figure 4.5: Example of a Petri net with inhibitor arcs

Definition 4.8 (enabledness—with inhibitor arcs)
For Petri nets with inhibitor arcs, a transition $t \in T$ is enabled by a marking M, iff it is enabled by M according to definition 4.4 and all its inhibiting places are empty. We will denote by ${}^\circ t$ the set of places that inhibit t, that is ${}^\circ t = \{p \in P \mid \exists (p,t) \in {}^\circ A\}$ We define the set of enabled transition by a marking M (in this context) as:

$$\mathsf{enab}(M) \triangleq \{t \in T \mid \forall p \in {}^\bullet t, M(p) > W(p,t) \wedge \forall p' \in {}^\circ t, M(p') = 0\}.$$

◊

Figure 4.5 gives an example of a Petri net with 2 inhibitor arcs. They forbid firing of transitions t_0 and t_1 when p_2 contains a token. In this net, transitions t_0 and t_1 cannot fire from marking $M = \langle 1,1,1,0 \rangle$. Inhibitor arcs can be used to represent block constraints in urban train sytems, i.e. impose that a train cannot enter an already occupied track portion.

Remark 4.3. (expressiveness) Adding inhibitor arcs to Petri nets gives them the expressive power of Turing machines. As a consequence, all the properties that are decidable for Petri nets (termination, boundedness, coverability, reachability) are undecidable for nets with inhibitor arcs.

When more than one transition is enabled by a marking M, only one transition can fire. Every transition can fire from M, but standard Petri nets do not specify the probability of firing of each transition. Similarly, Petri nets do not address time. Several timed and stochastic variants have been proposed. In the next sections, we present timed and stochastic variants that were relevant for the definition of the models introduced in Chapter 5.

4.2.2 Time Petri nets

In rail systems, time is an important factor. Train trips have nominal durations but are also subject to fluctuations due to some delays. These delays emerge randomly from different sources, e.g., technical failures, accidents, door-holding, etc. Guaranteeing safety and comfort of passengers calls for a proper reaction to these delays and for a proper modeling that integrates time.

There exist two major families of temporal extensions of Petri nets: time Petri nets (TPN) [52, 42] and timed Petri nets (TdPN) [59, 38]. Time Petri nets attach time constraints to transitions, and enforce firings of transitions to occur within a certain time interval after their enabling. Timed Petri net attach ages to token, and constraints to arcs. A transition is allowed to fire if the constraints on arcs from the preset of the transition to the transition are met. Coverability, termination and boundedness are decidable for timed Petri nets [1], but undecidable for time Petri nets. Reachability is undecidable for both models. In this section, we will mainly consider time Petri nets [52] and their variants.

Let us denote by $\mathbb{I}^+$ the set of intervals with positive rational endpoints; i.e., all intervals of the form $[\alpha,\beta]$, $[\alpha,\beta)$, $(\alpha,\beta]$ and (α,β), with $\alpha \in \mathbb{Q}_{\geq 0}$, $\beta \in \mathbb{Q}_{\geq 0} \cup +\infty$ and $\alpha \leq \beta$.

Definition 4.9 (time Petri net)
A *time Petri net* is a tuple $\langle P, T, A, W, M_0, I_s \rangle$ where P, T, A, W and M_0 are places , transitions, arcs, and an initial marking, $I_s : T \to \mathbb{I}^+$ is a function that associates a *static time interval* to transitions. ◊

The endpoints of the interval $I_s(t)$ are usually called the (static) *earliest* and *latest firing time* of transition t. They are respectively denoted by $\mathsf{eft}_s(t)$ and $\mathsf{lft}_s(t)$.

Semantics

A TPN can be seen as a classical Petri net in which execution times of transitions are restricted, i.e., a Petri net is a particular case of a TPN in which all transitions are associated with the interval $[0, +\infty)$. The principle of TPNs semantics is that a transition fires when it has been enabled for a duration $d \in I_s(t)$. Hence markings are not sufficient to take time constraints into account, and the semantics of TPNs is described in terms of discrete and timed moves from a *configuration* to the next one. Configurations memorize place contents, but also the time elapsed since enablings of transitions.

Definition 4.10 (configuration—of a TPN)
A configuration (state) of a TPN is a pair $\langle M, \nu \rangle$ where:

- M is a marking, and
- $\nu : \mathsf{enab}(M) \to \mathbb{R}^+$ is a clock value.

◊

The execution of a TPN can be described in terms of timed moves, that let a certain duration elapse, and discrete moves, that fire transitions and move tokens according to a standard Petri net semantics. Function ν recalls the time that has elapsed since the last enabling of each enabled transition during an execution of the net. We denote by $\nu + d$ the clock valuation that associates value $\nu(t) + d$ to transition t. A *timed move* from a

configuration (M, ν) to a configuration $(M, \nu + d)$ can occur, iff, for every enabled transition t, $\nu(t) + d \leq \mathsf{lft}_s(t)$. This type of move is denoted by $(M, \nu) \xrightarrow{d} (M, \nu + d)$

A discrete move from a configuration (M, ν) to (M', ν') is performed by firing a transition t. This discrete move is allowed only if t is enabled by M , and $\nu(t) \in I_s(t')$. The marking M' obtained is the usual marking in the untimed Petri net semantics. Last, the valuation ν' is computed as follows:

$$\forall t' \in \mathsf{enab}(M') : \nu'(t') = \begin{cases} \nu(t) & \text{if } t' \in \mathsf{pers}(M, t) \\ 0 & \text{if } t' \in \mathsf{newl}(M, t) \text{ or } t' = t \\ \text{is undefined} & \text{otherwise} \end{cases}$$

Calculation of the new firing intervals is done as follows:

$$\forall t' \in \mathsf{enab}(M') : I'(t') \triangleq \begin{cases} [\max(0, \mathsf{eft}(t') - d), \mathsf{lft}(t') - d] & \text{if } t' \in \mathsf{pers}(M, t) \\ I_s(t') & \text{if } t' \in \mathsf{newl}(M, t) \end{cases}$$

An important feature of time Petri nets is *urgency*. Time cannot elapse if there exists an enabled transition t such that $I_s(t) = [\alpha, \beta]$ (or $I_S(t) = (\alpha, \beta]$) and $\nu(t) = \beta$. Intuitively, time cannot elapse if a transition has been elapsed for a duration that reaches the highest possible duration allowed by $I_s(t)$. In this case, t or a transition that is enabled in (M, ν) must fire before time elapses. This rule give time Petri nets their expressive power. From a more pragmatic point of view, urgency allows for the specification of situations in which events have to occur before a certain date. In the context of UTS, this allows for instance to specify that a train must leave a station at latest two seconds after the departure order was given. Urgency is hence an important feature in the context of this thesis.

Time Petri nets allow for the design of models with clocks, delays, urgency, ... However, one can notice that randomness is still missing. Indeed, when several transitions are firable from a configuration, TPNs do not affect any probability to these transitions. Similarly, a transition t can remain enabled for the whole interval $[0, \mathsf{lft}_s(t)]$ and fire after a duration comprised in $I_s(t)$, but this dense space of legal durations is not probabilized.

4.2.3 Stochastic Petri nets

Stochastic Petri nets [53, 7] bring randomness in Petri nets. More precisely, possible firing times of transitions in this model are exponentially distributed. As a result, instead of knowing in which interval of time a transition is expected to fire, we know what is the probability of a transition firing before (or after) given times.

Definition 4.11 (stochastic Petri net)
A stochastic Petri net (SPN) is a tuple $\langle P, T, A, M_0, \lambda \rangle$ where:

- P, T, A, and M_0 form a Petri net, and
- $\lambda : T \to \mathbb{Q}_{\geq 0}$ a function that, to each transition $t \in T$, assigns a *rate*.

$\Diamond$

If we denote by ζ_t the random variable representing the time-to-fire of a given transition $t \in T$, then ζ_t is exponentially distributed and its CDF is given by $F_{\zeta_t}(x) \triangleq 1 - e^{-\lambda(t) \cdot x}$.

When a transition t is enabled, the average time for it to fire is given by $1/\lambda(t)$. If more than one transition are enabled by a marking M, then the probability for each enabled transition t to fire first is equal to $\lambda(t)/\sum_{t' \in \mathsf{enab}(M)} \lambda(t')$.

There exists an extension of SPNs called generalized stochastic Petri nets (GSPN) [3, 7]. In this variant, the set of transitions is composed of *timed transitions* that fire after a random exponentially distributed enabling time, as in SPNs, and *immediate transitions* that fire in zero time once enabled. When multiple immediate transitions are firable at the same time, the transition that fires is chosen according to a probability determined by weights of enabled immediate transitions. Finally, when both immediate and timed transitions are firable at the same time, only immediate transitions can fire.

4.2.4 Stochastic time Petri nets

In the model of SPNs/GSPNs, random variables are exponentially distributed. This choice facilitates analysis, but is too restrictive in many situations. For instance, the time needed to move from one station to another is not exponentially distributed, and is rather close to a Weibull distribution. SPNs do not allow to model many other stochastic systems whose random variables are not necessarily exponentially distributed. For this reason, VICARIO et al. [13, 39] propose the extension of STPNs.

Definition 4.12 (stochastic time Petri net)
A stochastic time Petri net is a tuple $\langle P, T, A, {}^\circ A, M_0, I_s, \mathscr{F}, \mathscr{W} \rangle$ where:

- P, T, A, ${}^\circ A$, M_0, and I_s form a (basic) time Petri net with inhibitor arcs;
- $\mathscr{F} : T \to \Sigma_{\mathsf{cdf}}$ associate each transition $t \in T$ with a CDF denoted $F_t \triangleq \mathscr{F}(t)$ with $\mathsf{dom}(F_t) = I_s(t)$; and
- $\mathscr{W} : T \to \mathbb{R}_{>0}$ associate each transition with a *weight*.

◊

For convenience, we will only assign closed intervals and open intervals of the form $[\alpha, +\infty)$ to transitions of stochastic time Petri nets. For each transition $t \in T$, let ζ_t denote the random variable with CDF F_t, and let $\{\zeta_{\langle t,0 \rangle}, \zeta_{\langle t,1 \rangle}, \zeta_{\langle t,2 \rangle} \ldots\}$ denote the set of (possibly infinite) outcomes of a random sampling experiment from $I_s(t)$ according to F_t.

Semantics

Definition 4.13 (configuration—of an STPN)
A configuration of a STPN is a tuple $\langle M, \tau \rangle$ where:

- M is a marking; and
- $\tau : \mathsf{enab}(M) \to \mathbb{R}_{\geq 0}$ associates each enabled transition with a time-to-fire.

◊

Intuitively, $\tau(t)$ is the time that t has to wait before being fired or disabled. In STPNs, a transition can fire if it is enabled, and its time to fire is equal to 0. Given a configuration $\langle M, \tau \rangle$, we define the set of firable transitions as:

$$\mathsf{fira}\big(\langle M, \tau \rangle\big) \triangleq \{t \in \mathsf{enab}(M) \mid \tau(t) = 0\}.$$

The semantics of an STPN is then defiend in terms of a *time move* or a *discrete move*.

A **discrete move** from a configuration $\langle M, \tau \rangle$ consists in firing a firable transition $t \in \mathsf{fira}(\langle M, \tau \rangle)$. This firing leads to a configuration $\langle M', \tau' \rangle$. M' is obtained following the regular firing rule of Petri nets (cf. Def. 4.5)

New times-to-fire are calculated as follows:

$$\forall t' \in \mathsf{enab}(M') : \tau'(t') \triangleq \begin{cases} \tau(t') & \text{if } t' \in \mathsf{pers}(M, t) \\ \zeta_{\langle t,i \rangle} & \text{if } t' \in \mathsf{newl}(M, t) \text{ or } t' = t \end{cases}$$

Here, $\zeta_{\langle t,i \rangle}$ is a value sampled randomly from $I_s(t)$. In practice, this sampling can be performed with the inverse transform sampling method.

A **time move** from a configuration $\langle M, \tau \rangle$ is performed by letting time elapse by a strictly positive duration $\theta \leq \min_{t \in \mathsf{enab}(M)} \tau(t)$, and leads to a new configuration $\langle M', \tau' \rangle$.

In time moves, the marking stays unchanged, i.e., $M' \triangleq M$, and times-to-fire are reduced by the elapsed time:

$$\forall t' \in \mathsf{enab}(M') : \tau'(t') = \tau(t') - \theta.$$

An execution of an STPN consists in a succession of time and discrete moves, e.g. a run of the form:

$$\langle M, \tau \rangle \xrightarrow{t_x} \langle M', \tau' \rangle \xrightarrow{\theta_0} \langle M'', \tau'' \rangle \xrightarrow{\theta_1} \dots \xrightarrow{t_y} \langle M^{(k)}, \tau^{(k)} \rangle \xrightarrow{t_z} \dots$$

Following this semantics, from any configuration only one type of move can be performed, i.e., when one transition (or more) is firable, only a discrete move can be performed, and when time can elapse, no transition is firable and only time moves can be performed.

When a time move is possible, an infinite number of time moves can theoretically be carried out. However, simulations usually perform the greatest possible time move with $\theta = \min_{t \in \mathsf{fira}(\langle M, \tau \rangle)} \tau(t)$. This approach allows for a considerable gain in execution time when using STPNs for simulation, in comparison with approaches in which time is discretized using a fixed period.

In some cases, more than one transition may be firable at the same time. In this case, transition weights are used, and the probability of firing a firable transition t is equal to $\mathcal{W}(t) / \sum_{t_i \in \mathsf{enab}(M)} \mathcal{W}(t_i)$.

4.2.5 Batches Petri nets

Certain variants of Petri nets (e.g., classical, time and stochastic variants) represent moving objects as tokens. These are discrete entities that move along a network. To model continuous quantities such (e.g., fluids), other models were later proposed, such as: continuous Petri nets (CPN) [21], in which places contain continuous quantities instead of tokens, and transitions consume place contents at a given rate, hybrid Petri nets (HPN) [4], that allow for the representation of continuous and discrete quantities in the same model; and later, batches Petri nets (BPN) [24], that was designed specifically for the representation of systems with circulating parts on conveyor belts and accumulation phenomena. Interestingly, the generalized version of this model, namely generalized batches Petri

nets (GBPN) [22, 23], allows for the modeling and analysis of high speed systems, and for the synthesis of control policies.

We refer interested readers to [22, 23] for a complete description of the model and of its semantics. In this section,we will mainly give an idea of the element appearing in a batch Petri nets.

Definition 4.14 (generalized batches Petri net)
A GBPN is a tuple $\langle P, T, A, W, \mathsf{type}, \mathsf{char}, \mathsf{mark}_0, \mathsf{temp}\rangle$ in which:

- P, T, A, and W are Petri net places, transitions, arcs and multiplicity function;
- $\mathsf{type} : P \cup T \to \{D, C, B\}$ associates each node (p or t) with a type: *discrete* (D), *continuous* (C) or *batch* (B);
- $\mathsf{char} : P \cap \mathsf{type}^{-1}[\{B\}] \to \mathbb{R}^3_{\geq 0}$ associates each batch place p_i with a triple of characteristics: $\langle \mathrm{vel}_i, \mathrm{den}_i, \mathrm{lgt}_i\rangle$ for *driving speed*, *maximum density*, and *length*;
- mark_0 is the initial marking; and
- $\mathsf{temp} : T \to \mathbb{Q}_{\geq 0}$ associates each transition t with a nonnegative rational number.

◊

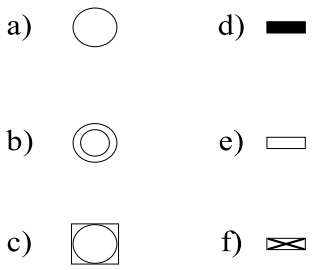


Figure 4.6: Elements of batches Petri nets: a) discrete places b) continuous places c) batch places d) discrete transitions e) continuous transitions f) batch transitions

Figure 4.6 lists elements that appear in batches Petri nets. Discrete places a) and discrete transitions d) have the same meanings as in Petri nets. Discrete places can contain tokens, that are moved by discrete transitions. Continuous places b) contain real quantities, that can be consumed by continuous transitions e) or batch transitions f. Continuous places can represent tanks, energy levels,... Last, batch places contain batches, i.e. loads of matter that are conveyed from the beginning of the place to its end in batches. Batch transitions consume and produce the contents on continuous places and of batch places. Batch places are filled and emptied by continuous and batch transitions.

A batch place can be seen as a conveyor, with a length and a speed. A batch is a load of matter with a density, a length and a position. Batch places contain several batches, that all move toward the end of the place at the same speed. Batch transitions consume the contents of their input continuous or batch places at some rate. They can be seen as machine processing A marking of a GBPNs associates to each discrete place a integer (number of tokens), to each continuous place a real value, and to each batch place a sequence of batches.

The evolution of places contents follows a mixed discrete/ continuous mode: discrete transition move tokens among discrete places, and batch/continuous transitions modify the contents of batch/continuous places at some rate when their are enabled. Controlling a

batch Petri net consists in controlling discrete transitions, and in fixing the rate of continuous and batch transitions.

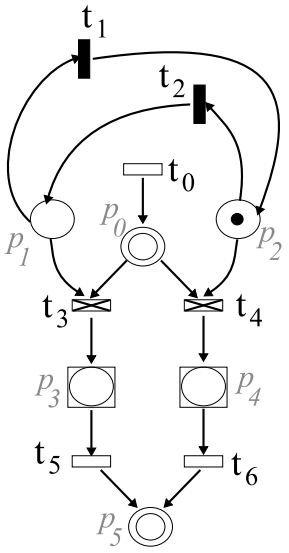


Figure 4.7: An example system modeled with a batches Petri nets

The example of Figure 4.7 is a simple plant model processing arrival with two machines. Quantities of matter to manufacture are produced by transition t_0 in place p_0. The contents of place p_0 can be sent to batch p_3 by transition t_3 when place p_1 is filled and sent to batch p_4 by transition t_4 when place p_2 is filled. Last transitions t_5 and t_6 release the contents of batches in place p_5. In this type of model, transition t_1 has a particular production rate, and by playing with the speed of batches and with transitions t_1 and t_2, one can control at which speed the contents of place p_0 is transferred to place p_5.

Batch Petri nets allow for the simulation of a wide range of DESs. Although batch elements and related mechanisms were initially designed to model production systems with accumulation phenomena, GBPNs can effectively be used to design control policies for transportation networks [23]. Batch places can be seen as portions of highways, with configurable speed constraints as a means of flow decongestion. It is to be noted that this model is at least as expressive as CPNs and HPNs as the sets of batch, continuous or discrete places and transitions can be empty, leading to instances of the aforementioned models. Batch mechanisms allowed for simulation of a portion of Netherlands' A12 highway and performance evaluation of the used speed control policies [23] using the A12segment tool [65].

Despite their expressiveness, batches Petri nets miss a few ingredients to model railway systems. Indeed, UTS need to consider random delays, which cannot be modeled in batches or continuous places/transitions, that have deterministic behaviors governed by the chosen rates of transitions. Another mismatch between UTS and batch Petri nets is that trains are discrete elements with a certain speed and length, and cannot be considered as a splittable flow of matter. Further, batch places cannot be used to model track in their current configurations: the contents of a batch place move at a unique speed, while trains have their own speed.

Despite the expressive power and appealing features of stochastic time Petri nets and batch Petri nets, these models are not sufficient to model regulated urban train systems.

For this reasons two new variants of stochastic time Petri nets were modeled. They are described in chapter 5.

4.2.6 Queuing networks

In order to optimize the use of capacity and traffic flow, several models for stochastic train operation were proposed [32]. One of the pioneering efforts in this field of research lay in the queuing theory based approach proposed by SCHWANHÄUSSER [62]. It aims at estimating the secondary delay following disturbances in a rail system. This delay corresponds to the mean queue length and is function of the distribution of primary delay, buffer time, mean headway, train sequence, and priority. The chosen model is of type $M/D/1$, i.e., arrivals are determined by a Poisson process (**M**arkovian process), jobs service times are fixed (**D**eterministic), and with 1 server. The admitted level of hinder is defined empirically by the maximum queue length which equals the maximum total secondary delay per day. It is given by the equation: $L_W = \alpha \cdot e^{-\beta \cdot pP}$ where: L_W is the queue length, α and β are coefficients, and pP probability of passenger trains. Operation quality is assessed by dividing the estimated queue length by the maximum queue length with 0.5 as a result standing for very good, 1 for acceptable, and 1.5 unsatisfactory quality.

The estimation of scheduled and unscheduled waiting times as a measure of operations' quality of a timetable is implemented in the software ANKE (German abbreviation for Analytic Network Capacity Determination) tool, developed at RWTH Aachen [67].

Due to the restrictive aspect of the $M/D/1$ queue, models of type $G/G/1$ (with **G**eneral, i.e., arbitrary distributions) and $M/G/1$ were later proposed by WAKOB [68] and HERTEL [36] respectively.

5

Models for regulated metro systems

This chapter proposes two models for urban train systems. These models are variants of Stochastic Time Petri nets introduced in Chapter 4. As already explained in Chapter 4, standard models of Petri nets miss some elements to represent features of urban train systems. In this chapter, we extend STPNs and define two models: the first one, introduced in Section 5.2 is a variant of STPNs with asymmetric distributions and a blocking semantics. In this model, trains are represented by tokens. The model is well adapted to represent URSs with fixed block policies. The second model, introduced in section 5.3 is also a variant of STPNs, but where trains are represented by trajectories. This model was inspired by STPNs, continuous nets and Batch Petri nets, but with some differences that are highlighted in the section. This model is well adapted to represent URSs with moving block policies. Both models rely on particular random distributions described in section 5.1

5.1 Random deviations

One of the first requirements in models for urban train systems is the capacity to faithfully represent the delays occurring during trips and at dwell times. To this extent, we first discuss the general shape of probability distributions that will be used in our models to represent delays.

During a normal operation day, train departures and arrivals are planned at certain known dates, usually specified in a timetable. However, these dates are rarely perfectly met, due to various disturbances delaying trains (doors problems, slight speed variations, weather conditions....) to which the system is exposed. Most of the time, events occur at dates that are slightly earlier or slightly later than the reference date specified in the timetable. Usually, these delays or advances are small (only a few seconds), and large differences between the planned date of an event and the real occurrence date are very rare.

In order to model this randomness, a natural approach is to consider dwell and running times as random variables following a probability law. The law depicting the duration of a dwell time or of a running time can be one of the off-the-shelf distributions (exponential law, normal laws, uniform law) seen in Chapter 2. However, these distributions are not always adapted to durations appearing in Urban Train Systems. If the duration of a given operation o is represented by a continuous random variable X, then the mode of variable X (the most probable value) should represent the nominal duration value for the duration of this operation. For instance, if a train transit from station st_a to station st_b is expected to last δ time units, then δ should be the mode of the distribution depicting the duration of a trip from station st_a to station st_b. Obviously, this duration cannot be represented by an uniform law. Normal distributions are better candidates, but yet are still too symmetric to model delays and advances of trains. Indeed, if the duration of an operation o is represented by a normally distributed random variable, the probability of a delayed is equal to the probability of an advance. Experience shows that, in reality, the probability of delays is substantially higher than the probability of avdances (w.r.t the nominal duration specified in the system) and that advance is bounded (due to maximal speed of trains and to minimal dwell times). This intuition was confirmed by analysis of three months of logs for the line 1 of Santiago's metro, showing asymmetric histograms. Figure 5.1 shows a typical distribution of trip durations between tow chosen consecutive stations in Santiago's Metro line 1. Obviously, this histogram does not coincide with a normal law.

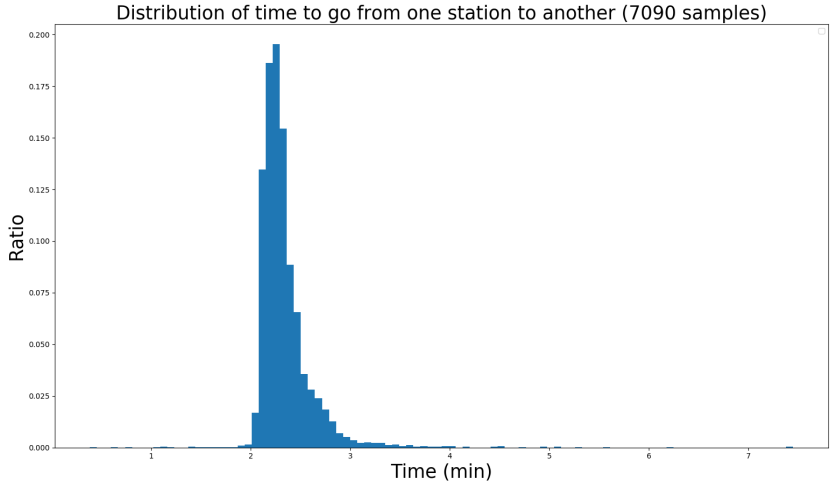


Figure 5.1: A histogram for the duration of trips between two stations of Santiago's Metro line 1 (3 months of logs)

Truncated expolynomial functions [39] are good candidates for specification of random durations in URSs. Truncated expolynomial functions compose several exponential laws, and can take positive values only on an given interval $[\alpha, \beta]$. The curves drawn for these functions are asymmetrical truncated bell-shaped curves. A PDF f defined with a trun-

cated expolynomial function is described as follows:

$$f(x) = \begin{cases} \sum_{i=1}^{n} c_i \cdot x^{a_i} \cdot e^{-\lambda_i \cdot x} & \text{if } x \in [\alpha, \beta] \\ 0 & \text{otherwise.} \end{cases}$$

with $c \in \mathbb{N}$, $a \in \mathbb{R}$, and $\lambda \in \mathbb{R}_{\geq 0}$.

When $f(x)$ defines a PDF of a random variable, its integral over its domain has to be equal to 1.

$$\int_{-\infty}^{+\infty} f(x)\,\mathrm{d}x = \int_{\alpha}^{\beta} f(x)\,\mathrm{d}x = 1.$$

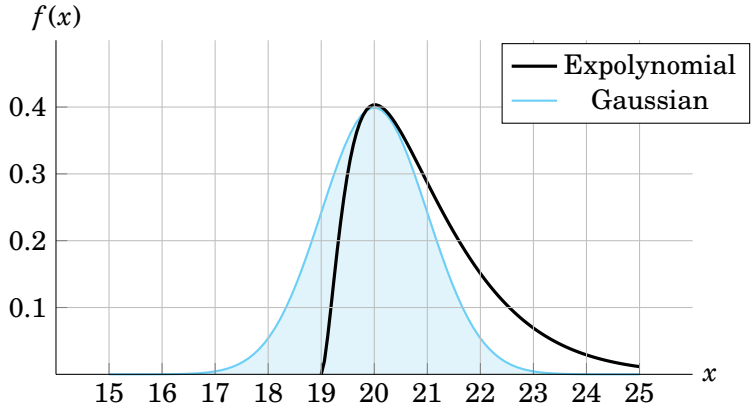


Figure 5.2: Comparison between Gaussian and truncated expolynomial functions

Figure 5.2 illustrates the advantage of using truncated expolynomial functions. The curves represented here are the Gaussian function $\mathcal{N}(20,1)$ in the background, and the function $d(x) = 19 + f(x)$, where $f(x)$ is the expolynomial function $6.20 \cdot x^{2.95} \cdot e^{-3.00 \cdot x} + 0.30 \cdot x^{1.50} \cdot e^{-1.20 \cdot x}$ defined on interval $[0,6]$. $\mathcal{N}(20,1)$ is represented by the blue area in the background, and $d(x)$ with a thick line. One can notice that $f(x)$ could be obtained with a Weibull function. However, truncated expolynomial functions allow for the definition of distributions with several local maximal probabilities (peaks in the curve), depicting for instance, the most probable duration for a dwell time, and the second most probable duration when an incident occurs as doors close.

In the rest of the chapter, we will use truncated expolynomial functions to model dwell and running times and their perturbations.

5.2 Stochastic time Petri nets with blocking semantics

In this section, we propose a model for URS. This model uses a variant of Stochastic Time Petri nets introduced in chapter 4, a timetable, and a regulation algorithm that recomputes timetables. To encode fixed block signaling systems, we define a particular blocking semantics, i.e. an elementary semantics of nets that forbids a train to enter a section that is already occupied by a train. We model the effects of regulation with particular control places, that are ignored to consider transitions as enabled, but must be filled if a transition needs to fire. These places serve as interface between a timetable, that dictates possible dates of departure, and the physical system descried by the blocking STPN.

The semantics of our model is a synchronized execution of a Stochastic Petri net and of a timetable that is constantly updated by a rescheduling algorithm. We detail the semantics of each part of the model, and then show how the net and the timetable synchronize.

5.2.1 Semantics of stochastic Petri nets

We recall that a STPN is a tuple $\langle P, T, A, {}^{\circ}A, M_0, I_s, \mathscr{F}, \mathscr{W} \rangle$. The blocking semantics of stochastic Petri nets consists in discrete transitions firings and timed moves.

Definition 5.1. A transition t is *enabled* in a marking m if and only if there is at least one token in each of its input places, and none of it inhibiting places is marked. We denote by $\mathsf{enab}(m)$ the subset of transitions enabled by a marking m and define it as:

$$\mathsf{enab}(m) = \{t \in T \mid \forall p \in {}^{\bullet}t, m(p) > 0 \wedge \forall p \in {}^{\circ}t, m(p) = 0\}$$

The semantics of TPNs is expressed in terms of moves between configurations. Intuitively, a configuration remembers the localization of trains, and the time remaining before a departure or an arrival of a train. Discrete transitions change markings (i.e. change the localization of trains) and timed moves let time elapse, hence decreasing the remaining time before a departure or an arrival.

Definition 5.2. A *configuration of a STPN* is a tuple $\mathscr{C}_N = \langle m, \tau \rangle$ where

- m is a marking of the STPN,
- $\tau : \mathsf{enab}(m) \longrightarrow \mathbb{R}_{\geq 0}$ is a function that assigns to each transition t_i enabled by marking m a positive real value $\tau_i = \tau(t_i)$ that represents the time left to fire transition t_i.

Definition 5.3. A transition t is *firable* in a configuration $\mathscr{C}_N$ if it is enabled by the marking of the configuration m, its time to fire $\tau(t)$ is equal to 0, and all places in the postset of t are empty.

This definition of firability enforces an elementary semantics. In the rest of the section, we will denote by $\mathsf{fira}(\mathscr{C}_N)$ the set of firable transitions in configuration $\mathscr{C}_N$. For a given $\tau : T \to \mathbb{R}_{\geq 0}$ and a value $\delta \in \mathbb{R}_{>0}$, we define the mapping $\tau - \delta$ as the mapping that associates $\tau(t) - \delta$ to any transition t in $Dom(\tau)$.

To avoid train collisions, signals forbid trains to enter a section that is already occupied by another train. In our net model, this will block firing of some transitions.

Definition 5.4 (blocked transition)**.** A transition t is *blocked* in a configuration $\langle m, \tau \rangle$ if and only if it is enabled by m, its time to fire τt is equal to 0, and one of its postset places contains a token. The set of blocked transitions in a configuration $\langle m, \tau \rangle$ is formally defined as follows:

$$\mathsf{blck}(\langle m, \tau \rangle) = \{t \in \mathsf{enab}(m) \mid \exists p \in t^\bullet, m(p) > 0 \wedge \tau(t) = 0\}$$

Timed moves

A timed move from a configuration $\langle m, \tau \rangle$ to another configuration $\langle m, \tau' \rangle$ consists in letting a strictly positive duration δ elapse. To be allowed, δ must be positive and smaller than all times to fire associated by τ to any transition enabled but not blocked in m. After a timed move, the new configuration reached is $\langle m, \tau' \rangle$, i.e. times to fire of all enabled transitions are decreased by δ time units. Blocked transitions keep their time to fire set to 0. This can be formally written as follows:

$$\frac{\begin{array}{ll} & \delta > 0 \\ \wedge & \forall t \in \mathsf{enab}(m) \setminus \mathsf{blck}(\langle m, \tau \rangle), \tau(t) \geq \delta \wedge \tau'(t) = \tau(t) - \delta \\ \wedge & \forall t \in \mathsf{blck}(\langle m, \tau \rangle), \tau'(t) = \tau(t) \end{array}}{\langle m, \tau \rangle \xrightarrow{\delta} \langle m, \tau' \rangle}$$

If one wants time steps to be maximal, the rule becomes:

$$\frac{\begin{array}{ll} & \delta > 0 \wedge \delta = \min\{\tau(t) \mid t \in \mathsf{enab}(m) \setminus \mathsf{blck}(\langle m, \tau \rangle)\} \\ \wedge & \forall t \in \mathsf{enab}(m) \setminus \mathsf{blck}(\langle m, \tau \rangle), \tau(t) \geq \delta \wedge \tau'(t) = \tau(t) - \delta \\ \wedge & \forall t \in \mathsf{blck}(\langle m, \tau \rangle), \tau'(t) = \tau(t) \end{array}}{\langle m, \tau \rangle \xrightarrow{\delta} \langle m, \tau' \rangle}$$

Discrete moves

Blocking STPN also evolve through discrete moves, that is firing of transitions. Let m be a marking, and let $t \in \mathsf{enab}(m)$. We define as usual the temporary marking $m_{tmp} = m - {}^\bullet t$. We say that a transition t_i is *newly enabled* after firing of transition t from m if $t_i \in \mathsf{enab}(m')$ and either $t_i = t$ or $t_i \notin \mathsf{enab}(m_{tmp})$. Similarly, we will say that a transition t_i is *persistent* after the firing of t from m if it is enabled in m and remains enabled in m_{tmp}. We denote by $\mathsf{newl}(m, t)$ the set of newly enabled transitions and by $\mathsf{pers}(m, t) = \mathsf{enab}(m_{tmp}) \cap \mathsf{enab}(m)$ the set of persistent transitions. For a transition $t_i \in \mathsf{newl}(m, t)$, a new time to fire in $[\mathsf{eft}(t_i), \mathsf{lft}(t_i)]$ is sampled according to the CDF F_{t_i} attached to transition t_i.

A discrete move from a configuration $\langle m, \tau \rangle$ to a configuration $\langle m', \tau' \rangle$ via firing of transition t is formally defined as the operational rule below:

$$\frac{\begin{array}{ll} & t \in \mathsf{fira}(\langle m,\tau\rangle) \\ \wedge & m' = m - {}^\bullet t + t^\bullet \\ \wedge & \forall t_i \in \mathsf{pers}(m,t), \tau'(t_i) = \tau(t_i) \\ \wedge & \forall t_i \in \mathsf{newl}(m,t), \tau'(t_i) \in [\mathsf{eft}(t_i), \mathsf{lft}(t_i)] \end{array}}{\langle m,\tau\rangle \xrightarrow{t} \langle m',\tau'\rangle}$$

One can notice that this operational rule allows any value in $[\mathsf{eft}(t_i), \mathsf{lft}(t_i)]$ for the time to fire of t_i. We can now address the probability distribution on discrete moves: When more than one transition is firable in a given configuration $\mathscr{C}_N$, each firable transition t_k has a probability to fire equal to $P_{\mathsf{fire}}(t_k)$, where:

$$P_{\mathsf{fire}}(t_k) = \frac{\mathscr{W}(t_k)}{\sum\limits_{t_i \in \mathsf{fira}(\mathscr{C}_N)} \mathscr{W}(t_i)}$$

Discrete moves represent departures or arrivals of trains. Sampled values are durations that are chosen for dwell times or trip times. From a given configuration $\langle m,\tau\rangle$ when transition t fires, denoting by $\zeta_i = \mathsf{samp}(F_{t_i})$ the value sampled for a newly enabled transition t_i, the probability to obtain ζ_i as time to fire for t_i is consistent with the CDF F_{t_i}, that is $P(\zeta_i < x) = F_{t_i}(x)$ for any $x \in [\mathsf{eft}(t_i), \mathsf{lft}(t_i)]$. Though no probability is explicitly mentioned in the semantics rules, the choice of a particular move, and the sampling of new values obey probability distributions. Hence the probability to move from $\langle m,\tau\rangle$ to $\langle m',\tau'\rangle$ is equal to the probability to fire transition t among all firable transitions, multiplied by the probability that samplings of times to fire for newly enabled transitions results in τ'.

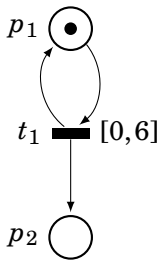


Figure 5.3: A simple STPN

Consider the example of Figure 5.3, and let us suppose that the net is in configuration $\langle m,\tau\rangle$, with $m(p_1) = 1$, $m(p_2) = 0$, $\tau(t_1) = 5.5$. This configuration can let 5.5 time units elapse, and then fire t_1. After this firing, a new time to fire d_1 is sampled for t_1, leading to a configuration $\langle m,\tau'\rangle$, where $\tau'(t_1) = d_1$.

Discrete departure orders

In the rest of this document, we will need to address marking changes due to the context of use of an STPN (namely departure orders sent by a timetable-based regulation scheme). A marking change is hence only a move from a configuration $\langle m,\tau\rangle$ to a configuration $\langle m',\tau'\rangle$ due to a change of marking from m to m' (departure orders will mainly add one tokens to a *control* place. These departure orders will be a way to synchronize the execution of a timetable and that of a STPN. For this reason, we need to redefine the notions of newly enabled and persistent transitions for such contextual marking changes.

Definition 5.5. Let m, m' be two markings. We will say that a transition t is persis-

tent with respect to contextual marking evolution from m to m' if t is enabled in m and m'. However, this marking change is not a firing of t, which forces us to redefine the notion of enabledness. We will say that t is newly enabled with respect to contextual marking evolution from m to m' if t is not enabled in m and is enabled in m'. We denote by $\mathsf{pers}(m,m') = \mathsf{enab}(m') \cap \mathsf{enab}(m)$ the set of persistent transitions after the evolution from m to m'. We denote by $\mathsf{newl}(m,m') = \mathsf{enab}(m') \setminus \mathsf{enab}(m)$ the set of newly enabled transitions after the evolution from m to m'.

The semantic rule defining a contextual change appending a token to place p is defined as follows:

$$\frac{\begin{array}{ll} & m' = m + \{p\} \\ \wedge & \forall t_i \in \mathsf{enab}(m), \tau'(t_i) = \tau(t_i) \\ \wedge & \forall t_i \in \mathsf{newl}(m,m'), \tau'(t_i) = \mathsf{samp}(F_{t_i}) \end{array}}{\langle m, \tau \rangle \xrightarrow{p} \langle m', \tau' \rangle}$$

Figure 5.4 illustrates how STPNs can be used to model a railway network, departures and arrivals. Here places p_j represents a station, and p_k represent two a track portion. Place p_{j-go} represents a departure order needed to fire transition t_{j-go} (the departure of a train. Place p_{j-out} is filled when a train has started to move from the station (but it has not yet effectively entered the next section). Transition t_j fires at the end of the nominal dwell time. A train enters the next section after firing of the transition t_k (the next section is represented by place p_k). For clarity, we have represented departure order with a gray token (as in place p_{j-go}), and trains with black ones (as in place p_j). However, the semantics of the STPN does not consider colors for tokens. To complete the picture, one can associate cumulative distributions to transitions. For transition t_{j-go}, this distribution represents the time needed to leave the station once the departure order is given by the timetable. This time can vary between 0 and 10 seconds, depending on the crowd in station, on door closure problems, etc... For transition t_j, t_k, t_l, the distribution symbolizes the trip time, that is the time needed to move from one section to the next one.

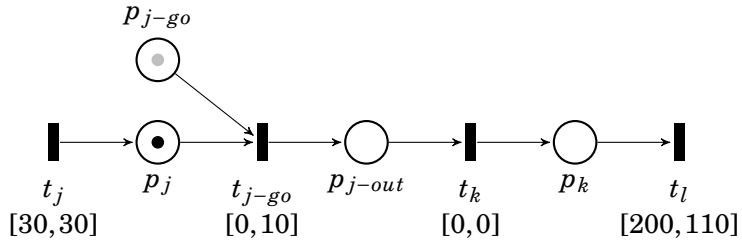


Figure 5.4: Transitions and places from a STPN model of a metro network.

5.2.2 Timetable Execution

The second ingredient of our model is a timetable. A railway system is driven according to a schedule, which is an *a priori choice* of for departures and arrival dates. This schedule is a purely theoretical view of the real system, in which delay and failures may impose changes to the planned schedule. We hence consider that during the execution of a timetable, departures or arrivals can occur at different dates than those originally planned. However, the operators use regulation policies to minimize the effect of delays,

and try to stick to the daily schedule.

As for Petri nets, we define configurations for timetables. A configuration of a timetable will have to remember which events have already been executed and their occurrence dates, the departure orders that have been sent, and the expected dates of remaining unexecuted events.

Definition 5.6. A *configuration* of a timetable $TH = \langle \mathscr{G}(S), \Delta_0 \rangle$ is a tuple $\mathscr{C}_T = \langle X, D, \Delta \rangle$ where

- $X \subseteq N$ is a set of nodes depicting the set of executed events from the timetable.
- D is a set of nodes that symbolize the departure orders that have already been sent to the corresponding trains, but did not necessarily occur.
- $\Delta : N \to \mathbb{Q}_{\geq 0}$ is a dating function that assigns an effective date to each node of X and an estimated date to each node $n_i \in N \setminus X$

The initial configuration $\mathscr{C}_T^0$ of a timetable TH is $\langle \varnothing, \varnothing, \Delta_0 \rangle$.

We represent departure orders that have been sent as a subset of departure events D. This set is essential to avoid filling twice a departure order place, and differentiate departures that are allowed (i.e. appear in D) and those that are executed (i.e. appear in X). Hence, in a configuration $\mathscr{C}_T = \langle X, D, \Delta \rangle$, D needs not be contained in X. This distinction between order and execution of the order allows to add delays at station due to doors or passengers problems.

Figure 5.5 is an example of a configuration, with 2 trains, and 4 events for each train. In this picture, gray nodes represent events that have already been executed ($X = n_{i1}, n_{j1}, n_{k1}, n_{i1}, n_{j2}$). White events represent events that are planned in the timetable, but have not yet been executed. For convenience, we also represent the last executed event as an event with bold font. The departure orders that have been sent are represented by doubly circled events, that is $D = \{n_{i1}, n_{i2}, n_{k1}, n_{k2}\}$. On this drawing, one can immediately notice that some black events ($n_{i1}, n_{i2} and n_{k1}$) are doubly circled: they correspond to departures that have been executed after the order was sent. The doubly circled event n_{k2} is not yet executed, however, an order corresponding to this departure has been sent to the trains.

Remark: for simplicity, we consider that departure orders are never lost and sent only once.

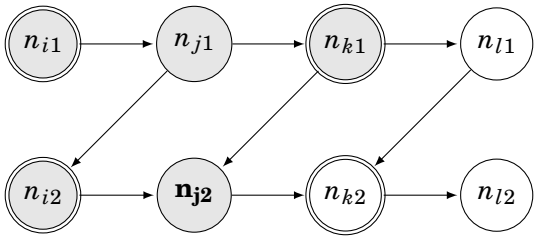


Figure 5.5: Configuration of a timetable

Let us recall that timetables dated graphs. A timetable hence gives an ordering on events. The main idea in timetable execution is to execute events in the prescribed order and at the chosen date in the timetable. To use the ordering imposed by a timetable, we first need to define the notions of predecessors and successors of an event in a timetable.

Definition 5.7. A *direct predecessor* of a node n_i is a node n_j such that (n_j, n_i) is an arc

of the timetable.A *direct successor* of n_i is a node n_j such that (n_i, n_j) is an arc of the timetable. We will denote by $d\sigma^-(n_i)$ the set of all the direct predecessors of node n_i and $d\sigma^+(n_i)$ the set of its direct successors.

Note that a node can have more than one direct predecessor (resp. direct successor) and can be the direct predecessor (resp. direct successor) of more than one node.

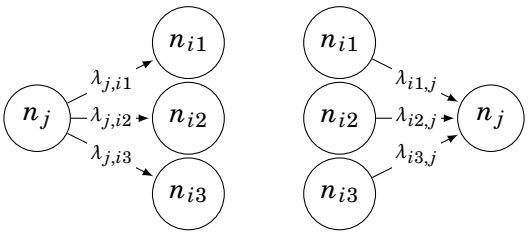


Figure 5.6: Direct predecessor and direct successor nodes

Definition 5.8. A node n_i is a *predecessor* (respectively a successor) of a node n_j in a timetable TT if there exists a sequence of nodes $(n_0, ..., n_k)$ such that $\forall 0 \leq i < k, (n_i, n_i + 1)$ is an edge of TT and $n_i = n_0 \wedge n_j = n_k$ (respectively $n_j = n_0 \wedge n_i = n_k$).

We will denote by $\sigma^-(n_i)$ (resp. $\sigma^+(n_i)$) the set of all predecessors (resp. successors) of node n_i.

Given a timetable $TH = \langle \mathcal{G}(S), \Delta_0 \rangle$, and a configuration $\mathcal{C}_T = \langle X, D, \Delta \rangle$, we will denote by $firstexec(TH, \mathcal{C}_T) = \{n \in N \setminus X \mid \nexists n', n \in E \cap (N \setminus X)^2\}$ the minimal set of nodes in the yet unexecuted part of the timetable. These nodes are the next events that can be executed from configuration $\mathcal{C}_T$. Similarly, we can define the next departure orders to be given, as $NextDep(TH, \mathcal{C}_T) = (firstexec(TH, \mathcal{C}_T) \cap N_d) \setminus D$. The next departure orders correspond to departure events that can be executed (preceding arrivals have already occurred) and for which a departure order was not already sent.

We are now ready to describe the execution of a timetable. Timetables semantics rules can be decomposed into timed moves (letting time elapse), orders, and actual execution observation. Execution of a semantic rule depends on the current date. The semantic rules follow the following requirements:

- departure orders are sent on time, according to the current schedule
- departures can not occur earlier than scheduled, but can be delayed
- arrival can occur earlier or later than expected.
- the system can simply let time elapse, but not up the the point where a departure order is forgotten.

The semantics rules hereafter recall a configuration $\mathcal{C}_T$ of a timetable and the current date d. They describe what happens when time elapses, or when an event occurs.If an event occurs at a date that differs from the expected date, then a re-scheduling of the remaining part of the timetable may have to occur. We describe this rescheduling as a function $h()$ that transforms a timetable into another timetable. This function considers the initial constraints in the graph, the set of executed events, the last event executed, and the current date to change dates of remaining events in the timetable.

Timed moves: From a configuration $\mathcal{C}_T = \langle X, D, \Delta \rangle$ at a date d, one can safely let δ time units elapse if the new date reached does not exceed the dates of next departure

orders.

$$\frac{d+\delta < \min(\Delta(NextDep(TH,\mathscr{C}_T)))}{\langle X,D,\Delta\rangle, d \xrightarrow{\delta} \langle X,D,\Delta\rangle, d+\delta}$$

departure order: A departure order is sent whenever the date to leave is reached. Note that this departure order can only be sent when the preceding arrival has already occurred, that is when the train receiving the order is in station.

$$\frac{\begin{array}{l} n \in NextDep(TH,\mathscr{C}_T) \\ \wedge d = \Delta(n) \end{array}}{\langle X,D,\Delta\rangle, d \xrightarrow{d(n)} \langle X,D\cup\{n\},\Delta\rangle, d}$$

early arrival: When a train arrives earlier than expected at a station, its arrival date is memorized.

$$\frac{\begin{array}{l} n \in firstexec(TH,\mathscr{C}_T) \cap N_a \\ \wedge d < \Delta(n) \end{array}}{\langle X,D,\Delta\rangle, d \xrightarrow{n} \langle X\cup\{n\},D,\Delta_{[n\to d]}\rangle, d}$$

where $\Delta_{[n\to d]}$ is the dating function that associates $\Delta(n')$ to any node $n' \neq n$, and d to node n[1]

Late arrival: When a train arrives at a station with a delay, several situations can occur. Either this delay can be absorbed by reducing the dwell time of the train, or there is a need to reschedule the next departure of this train, and of other causally dependent events. This rescheduling is made by a regulation algorithm, symbolized by function $h()$. We do not detail a particular rescheduling algorithm. The next subsection is devoted to rescheduling, and presents a simple delay propagation algorithm. The late arrival rule can be defined as follows:

$$\frac{\begin{array}{l} n \in firstexec(TH,\mathscr{C}_T) \cap N_a \\ \wedge d > \Delta(n) \end{array}}{\langle X,D,\Delta\rangle, d \xrightarrow{n} \langle X\cup\{n\},D,h(X,\Delta,d,n)\rangle, d}$$

Late departure

Late departure follows the same rule as late arrival, with the additional constraint that a departure order must have been given before the departure occurs.

[1] We do not consider cases where timetables are rescheduled when a train arrives early, but the rule can be adapted to reschedule the timetable.

$$\frac{\begin{array}{l} n \in firstexec(TH, \mathscr{C}_T) \cap D \\ \wedge d > \Delta(n) \end{array}}{\langle X, D, \Delta \rangle, d \xrightarrow{n} \langle X \cup \{n\}, D, h(X, \Delta, d, n) \rangle, d}$$

On time events

Events that occur at dates originally planned by the schedule do not require re-scheduling.

$$\frac{\begin{array}{l} n \in firstexec(TH, \mathscr{C}_T) \cap D \\ \wedge d = \Delta(n) \end{array}}{\langle X, D, \Delta \rangle, d \xrightarrow{n} \langle X \cup \{n\}, D, \Delta \rangle, d}$$

$$\frac{\begin{array}{l} n \in firstexec(TH, \mathscr{C}_T) \cap N_a \\ \wedge d = \Delta(n) \end{array}}{\langle X, D, \Delta \rangle, d \xrightarrow{n} \langle X \cup \{n\}, D, \Delta \rangle, d}$$

5.2.3 Regulation functions

In section 5.2.1, we have seen how to simulate physical infrastructure and random delays with a blocking variant of stochastic Petri net. We have defined runs of a system as prescribed by a timetable: events (departure or arrivals) occurs early, on time or late, and the timetable can send departure orders. Upon delayed event, the timetable can be recomputed using a a regulation function, implemented as a rescheduling algorithms. We now formalize rescheduling policies.

Departure order are given at dates specified by a timetable. Actual departures may occur later than expected, arrivals may occur earlier or later than expected. Delays may impose changes to the expected schedule, and these changes are performed according to a rescheduling policy. So far, we have let this rescheduling policy unspecified (it was simply given as a function $h(.)$ transforming a timetable into another timetable). We can now formalize these functions, and give an example of rescheduling function that compute the smallest allowed shift in a timetable.

Definition 5.9 (regulation function)**.** Let $\mathscr{T}$ denote the set of all timetables, $\mathscr{C}$ their possible configurations, and $\mathscr{G}$ denote the set of all alternative graphs. A *regulation function* is a function $\gamma : \mathscr{T} \times \mathscr{C} \times \mathscr{G} \times N \times \mathbb{R}_{>0} \to \mathscr{T}$.

We assume that the timetable that is executed is a solution obtained from a given alternative graph. Intuitively, a regulation function takes as input a timetable and its configuration, the alternative graph from which the timetable was computed, an event identity and its occurrence date. This is the event that caused the rescheduling. The output of a regulation function is a new timetable. As $h(.)$ may use an alternative graph, one can even design function that changes the order of events, for instance to reorder passages of trains at forks. We furthermore impose that regulation functions do not reschedule events originally planned before the occurrence date of the event causing the rescheduling.

Computing a new timetable may consist in a simple shift of dates, taking into account the constraints on dates of events inherited from the alternative graph. However, rescheduling may mean choosing a completely different set of alternative arcs for the yet unexecuted part of the alternative graph. In particular, it can allow to choose different routes for train, swap trains orders, etc.

Let us now define a simple regulation function γ that simply shifts the occurrence dates of events to the earliest date greater than the original schedule that satisfies all constraints on selected (fixed) arcs.

That is $\gamma(\Delta, C = (X, D, \Delta), k, d)$ is the solution Δ' of the set of equations

$$\begin{aligned}
&\forall x \in X, \Delta'(x) = \Delta'(x) \\
&\forall x \in N \setminus X, \Delta'(x) \geq \Delta'(x) \\
&\forall (x,y) \in E, \Delta'(y) - \Delta'(x) \geq \lambda(x,y)
\end{aligned}$$

A new earliest date for all unexecuted events can be easily obtained by a propagation algorithm such as algorithm 3.1 introduces in Chapter 4.

5.2.4 Scheduled simulation of a train system

The semantics of a regulated urban train systems is a joint simulation of a real system, represented as a Petri net, and of a timetable. We have seen in previous sections how to define individual semantics of Stochastic timed Petri nets, and of timetables. The two models are composed the following way:

- the timetable acts as a controller for departure transitions in the Petri net, by inserting tokens in specific departure places.
- the transitions of the STPN symbolize departures and arrivals. They correspond to nodes of the timetable, hence transitions of the STPN and nodes of the timetable are executed jointly in semantic rules. The execution date of an STPN transition is used to produce random changes with respect to the scheduled dates of events. Large delay may impose rescheduling of a timetable.

We now need to define how transitions, places, and events are mapped to allow joint simulation of two models. We fix a timetable $TH = \langle \mathscr{G}(S), \Delta_0 \rangle$, a Petri net STPN = $\langle P, T, L, H, m_0, \underline{T}, \overline{T}, \mathscr{F}, \mathscr{W} \rangle$, and a regulation policy h. In the rest of this section, we will denote by Σ_{tr} the set of trains and by Σ_{sc} the set of railways sections (track portions).

Among these sections we distinguish a particular set $\Sigma_{\text{st}} \subset \Sigma_{\text{sc}}$ of sections, that represent stations. Intuitively, a station is a section where a train has to stop and stay for a minimum dwell time. However, some track portions are not stations, and trains simply cross them without loading or unloading passengers. A train can leave the station when it receives a departure order (this is symbolized by a particular place that is filled according to orders coming from the timetable execution) and the next station is unoccupied by another train.

On the Petri net side, we will distinguish the set $Ta \subset T$ of arrival transitions, whose firing represent an arrival of a train at a section, and the set and $Td \subset T$ of departure transitions whose firing represent the beginning of the trip of a train from a track portion to another track portion. Note that we interpret the term "departure" as the fact that a

train has started to move, and not as the fact that the train has left the section. A train is considered to have effectively left the station only when it enters the next section of the line.

We let $P_{\text{arr}} \subset P$ denote a set of arrival places. An arrival place containing a token represents a train stopped at a station. We let $P_{\text{dep}} \subset P$ denote a set of departure places. A departure place containing a token is interpreted as a train on a trip to the next track portion. Let $Pc \subset P$ denote a set of departure control places used to give departure orders. Such departure places contain a token when a departure order was given (note that even if the departure order is given, the train need not move immediately).

The last ingredient of our model is a pair of mappings Ψ_t, ψ_c that allow to connect nodes of the timetable and places/transitions of the Petri net in our model. We let $\Psi_t : Ta \to \wp N$ denote a function that associates a set of arrival nodes to each arrival transition $t_{\text{arr}} \in Ta$, and a set of departure nodes to each departure transition $t_{\text{dep}} \in Td$. This map will be used to find which event of a timetable corresponds to the execution of a transition in the stochastic net. Similarly, departure orders scheduled from the timetable control the execution of the net by adding a token at a desired control place. The relation between departure orders of the timetable and control places of the stochastic net is defined as a map $\psi_c : Nd \to Pc$ that associates a departure control place to each departure node in the timetable.

As one can notice, several nodes of the timetable are attached to a transition of the Petri net. However, at a given instant d, considering the set of events X that have already been executed, one can easily find which event corresponds to a given transition t. We define the function $corr : 2^N \times T \to N$ as the mapping that indicates which node of the timetable corresponds to a given transition. More precisely, we set:

$$\mathsf{corr}(X,t) = \{n \in \Psi_t(t) \mid \nexists n' \in \Psi_t(t) \setminus X, n' \in \sigma^+(n)\}$$

As departures /arrivals from a chosen section should be ordered in the timetable, $\mathsf{corr}(X,t)$ should contain at most one element. Abusing the notation, we will write $n = \mathsf{corr}(X,t)$ to denote that $\mathsf{corr}(X,t) = \{n\}$.

A railway system is a pair (TT,N) composed of a timetable and of a Petri net. The overall semantics of railway system combines the execution of the schedule provided by timetable TT, which decides at which dates the departure orders must be sent to the physical network, and an execution of the Petri net N, which provides execution dates for arrival and departure transitions. These dates are subject to randomized variations with respect to the expected dates planned in the timetable. A configuration of a railway system (TT,N) is a triple $\langle \mathscr{C}_T, \mathscr{C}_N, d \rangle$, where $\mathscr{C}_T = \langle X, D, \Delta \rangle$ is a configuration of timetable TT, $\mathscr{C}_N = \langle m, \tau \rangle$ is a configuration of the Petri net N, and $d \in \mathbb{R}$ is the current date.

A joint simulation of a timetable and of a Petri net consists of timed moves, and discrete moves that are **allowed by both models**.

Timed moves:

Timed moves consist in letting time elapse if both the Petri net and the timetable allow it from their configurations at current date. A delay $x \in \mathbb{R}$ can elapse in a configuration $\langle \mathscr{C}_T, \mathscr{C}_N, d \rangle$ as soon as x does not exceed the maximal delay allowed by the timetable and by the Petri net. We avoid successive samplings of delays, and choose to let time elapse

until the unique earliest date $d+\delta$ of occurrence of an event. In other words, $d+\delta$ is equal to the earliest date of a departure order in the timetable, or of a transition firing in the STPN, and δ is the delay that elapses from current configuration. This way, time progresses until occurrence of the next event, which allows for efficient asynchronous and simulation. Time elapsing is described by the following semantic rule:

$$\frac{\begin{array}{l}\langle X,D,\Delta\rangle,d\rangle \xrightarrow{\delta} \langle\langle X,D,\Delta\rangle,d'\rangle \\ \langle m,\tau\rangle \xrightarrow{\delta} \langle m,\tau'\rangle \\ d' = d+\delta\end{array}}{\langle\langle m,\tau\rangle,\langle X,D,\Delta\rangle,d\rangle \xrightarrow{\delta} \langle\langle m,\tau'\rangle,\langle X,D,\Delta\rangle,d'\rangle}$$

This way, time progresses until occurrence of the next event, which allows for efficient asynchronous simulation.

Discrete moves

Several kinds of discrete moves can occur: a departure order imposed by the timetable, an arrival, or a departure decided by the Petri net, and expected from the current timetable configuration. The only technical point here is to relate departure orders with places of the Petri net that symbolize such orders, and transitions of the Petri nets with the corresponding event in the timetable. Then, the current date allows to decide whether the event occurs early, late, or on time. A discrete move hence consists in firing a transition or in appending a token in a control place.

Firing a transition t in the STPN means moving from a configuration $\langle m,\tau\rangle$ to a new configuration $\langle m',\tau'\rangle$. Now, this transition should correspond to a departure/arrival event scheduled in the timetable. Let $n_e = \mathsf{corr}(X,t)$ denote the next executable node that corresponds an execution of transition t in the timetable. Upon execution of t, node n_e is added to the set of executed nodes X. Dates in $\Delta(n)$ are updated: $\Delta(n_e)$ is defined, and takes as value the actual date d, dates of executed nodes X and nodes that are not a successor of n_e stay unchanged while dates of successors of n_e are recomputed according to the chosen regulation algorithm.

The semantic rule describing the discrete move caused by firing a transition (departure or arrival) is defined as follows:

$$\frac{\begin{array}{l}t \in Ta \cup Td \\ \langle m,\tau\rangle \xrightarrow{t} \langle m',\tau'\rangle \\ n = \mathsf{corr}(X,t) \\ \langle X,D,\Delta\rangle,d \xrightarrow{n} \langle X',D',\Delta'\rangle,d\end{array}}{\langle\langle m,\tau\rangle,\langle X,D,\Delta\rangle,d\rangle \xrightarrow{t} \langle\langle m',\tau'\rangle,\langle X',D',\Delta'\rangle,d\rangle}$$

Note : Δ' is the dating function that is recomputed during the timetable move.

When the next event occurring is a departure order (that is a discrete move imposed by the timetable, symbolized by a node n), we append a token in the departure control place $p = \psi_c(n)$ associated with node n. This leads to a new marking m' and a new TTF vector τ', as defined by the **contextual change rule** in the semantics of STPN seen earlier in this chapter. Similarly, the configuration of the timetable is changed according to the

departure rule. These changes are described by the following joint rules:

$$\frac{\begin{array}{l} p = \psi_c(n) \\ \langle m, \tau \rangle \xrightarrow{p} \langle m', \tau' \rangle \\ \langle X, D, \Delta \rangle, d \xrightarrow{d(n)} \langle X, D', \Delta \rangle, d \end{array}}{\langle \langle m, \tau \rangle, \langle X, D, \Delta \rangle, d \rangle \xrightarrow{p} \langle \langle m', \tau' \rangle, \langle X, D', \Delta \rangle, d \rangle}$$

Conclusions on blocking STPNs

The blocking STPNs presented in this section can model urban train systems with a fixed block policy, regulation schemes, and timetables. The way to build a SPTN from a metro topology is rather straightforward : blocks and stations are symbolized by places, transitions represent arrival, departures, ends of waiting times, transit to the next block,.... Overall, the STPN corresponding to a given topology can be obtained by assembling component of the form of the pieces of net in Figure 5.4.

The difficult part for this model is to build large timetables, and automate this process as much as possible, and then to establish the correspondences between transitions in the STPN and events in the timetable. This model was used to simulate an existing Metro line, namely Line 1 of Santiago's network. The results of the simulation campaign are presented and discussed in Chapter 7.

5.3 Trajectory Petri nets

STPNs with blocking semantics are a rough abstraction of fixed block systems, and cannot model systems with moving blocks. For the representation of moving block systems, we introduce trajectory Petri nets. The particularity of this model is that places contain trajectories instead of tokens. A trajectory in a place not only provides the current state of the modeled object, but also the expected changes in its position over time. Trajectories in this model are abstracted to a succession of segments. We provide their formal definition hereafter.

5.3.1 Trajectories

Definition 5.1 (point)
A *point* is a pair of *coordinates* $pt \triangleq \langle x, y\rangle \in \mathbb{R}_{\geq 0} \cup \{+\infty\} \times \mathbb{R}_{\geq 0}$. We define addition over points as: $\langle x_0, y_0\rangle + \langle x_1, y_1\rangle \triangleq \langle x_0 + x_1, y_0 + y_1\rangle$. ◊

Points are used to define segments, but, in some particular cases, a pair of points may represent a ray; in this case, the second point's abscissa is $+\infty$. Rays are necessary for the representation of trajectories of unknown duration. More details are provided in the semantics section for trajectory Petri nets.

Definition 5.2 (segment)
A *segment* is an ordered pair of points $sg \triangleq \langle pt_0, pt_1\rangle$ with the conditions:

$$\left[\begin{array}{lll} & x_0 \neq +\infty & \text{(first abscissa of segment has a finite value)} \\ \wedge & x_1 \geq x_0 \wedge y_1 \leq y_0 & \text{(segment has increasing abscissae and a decreasing ordinate)} \\ \wedge & x_1 = x_0 \implies y_1 = y_0 & \end{array}\right.$$

We say that a point $pt \triangleq \langle x, y\rangle$ *belongs to* a segment $sg \triangleq [\langle x_0, y_0\rangle, \langle x_1, y_1\rangle]$, and write $pt \in sg$, iff

$$\begin{cases} pt = pt_0 & \text{if } pt_1 = pt_0 \\ x \geq x_0 \wedge y = y_0 & \text{if } x_1 = +\infty \\ y = (x - x_0) \cdot \dfrac{y_1 - y_0}{x_1 - x_0} + y_0 & \text{otherwise.} \end{cases}$$ ◊

Definition 5.3 (trajectory)
A *trajectory* is an ordered set of consecutive segments $tj \triangleq \{sg_k \mid k = 0, 1, \ldots, Ns_j - 1\}$. $Ns_j \in \mathbb{N}$ is the size of trajectory tj, i.e., the number of its segments.

For convenience, given an ordered set of trajectories $\{tj_0, tj_1, \ldots, tj_{Nj-1}\}$, we write:

- $sg_{j,k}$ to denote the k^{th} segment of trajectory tj_j;
- $pt_{j,k,l}$ to denote the l^{th} point of segment $sg_{j,k}$; and
- $x_{j,k,l}$ and $y_{j,k,l}$ to respectively denote the abscissa and ordinate of point $pt_{j,k,l}$.

As segments of a trajectory must be consecutive, they satisfy : $\forall k = 0, 1, \ldots, Ns_j - 2 : pt_{j,k,1} \triangleq pt_{j,k+1,0}$. ◊

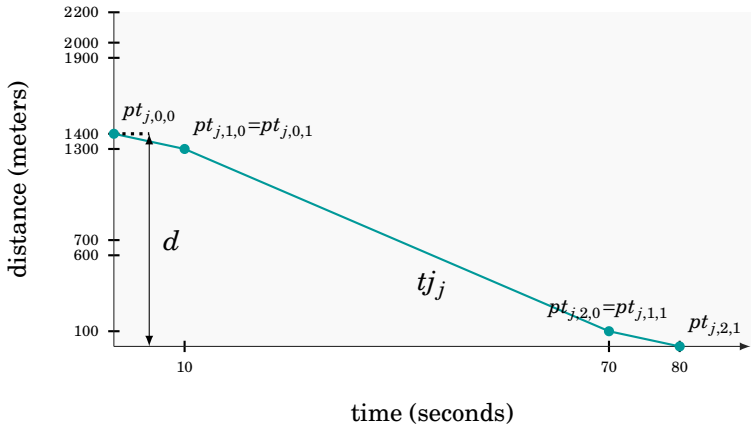


Figure 5.7: Example of a trajectory

Each trajectory tj_j can be seen as a function that defines distance to an arrival as a function of time, and maps abscissae (time) of its segments to ordinates (distance):

$$tj_j : \bigcup_{k=0}^{Ns_j-1} [x_{j,k,0}, x_{j,k,1}] \rightarrow \bigcup_{k=0}^{Ns_j-1} [y_{j,k,0}, y_{j,k,1}].$$

We will write $y = tj_j(x)$ to refer to the ordinate of a point $\langle x, y \rangle$ in a segment $sg_{j,k}$ of trajectory tj_j. We similarly write: $x = tj_j^{-1}(y)$ to refer to the abscissa corresponding to ordinate y in trajectory tj_j. If tj_j is not injective, we will abusively use the notation $tj_j^{-1}(y)$ to refer to the minimal abscissa such that that $tj_j(x) = y$.

We say that a trajectory $tj_j \triangleq \{sg_0, sg_1, \ldots, sg_{Ns_j-1}\}$ *covers a distance* $d \in \mathbb{R}_{>0}$ iff

- $pt_{j,0,0} = \langle 0, d \rangle$; and
- $y_{j,Ns_j-1,1} = 0$.

Figure 5.7 depicts an example trajectory with 3 segments covering a distance $d = 1400\ m$.

To define trajectory Petri nets and their semantics, we first need to introduce the notions of *upward shift*, *exclusion zone*, *blocking*, and *left shift* hereafter.

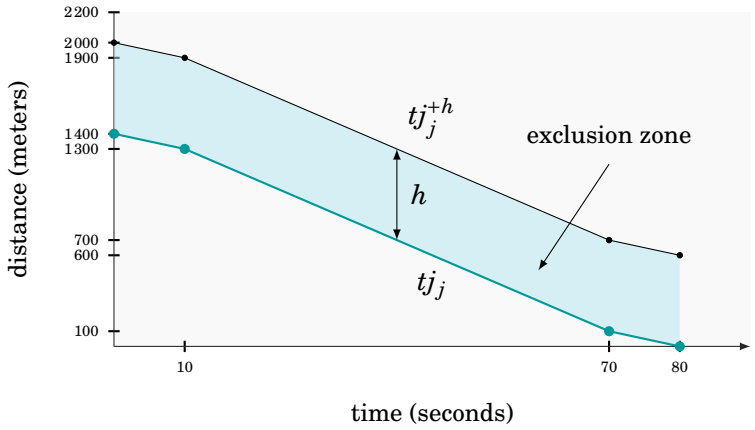


Figure 5.8: Upward shift and exclusion zone of a trajectory

Definition 5.4 (upward shift)
Given a trajectory tj_j and a headway $h \in \mathbb{R}_{>0}$, we define the *upward shift* of tj_j (w.r.t. headway h) as tj_j^h such that:

- $\mathsf{dom}(tj_j^h) \triangleq \mathsf{dom}(tj_j)$ (trajectories have the same abscissae); and
- $\forall x \in \mathsf{dom}(tj_j^h) : tj_j^h(x) \triangleq tj_j(x) + h$ (ordinates are increased by h). ◊

The upward shift of a trajectory is used to define an *exclusion zone* in a time-space diagram contained between tj_j and tj_j^h. If a train has a trajectory tj_j and is followed by another train with trajectory tj_i, then, if tj_i respects the safety headway h, it should not cross this exclusion zone. that should not contain a trajectory if headway h w.r.t. trajectory tj_j is respected. The upward shift can be easily computed by adding $\langle 0, h\rangle$ to every point in tj_j.

Figure 5.8 shows a trajectory, its upward shift by a headway $h = 600\ m$, and the exclusion zone they form.

Definition 5.5 (blocking)
Let tj_{j_0} and tj_{j_1} be two trajectories. We say that trajectory tj_{j_0} *blocks* trajectory tj_{j_1} w.r.t. some headway h iff:

- $tj_{j_0}(0) < tj_{j_1}(0)$ (tj_{j_1} is above tj_{j_0}); and
- $\exists x \in \mathsf{dom}(tj_{j_0}) \cap \mathsf{dom}(tj_{j_1}) : tj_{j_0}^h(x) > tj_{j_1}(x)$ (tj_{j_1} contains points below $tj_{j_0}^h$).

We say that trajectory tj_{j_0} blocks tj_{j_1} at abscissa $x \in \mathsf{dom}(tj_{j_0}) \cap \mathsf{dom}(tj_{j_1})$ iff:

- tj_{j_0} blocks tj_{j_1}; and
- x is the minimal abscissa the satisfies $tj_{j_0}^h(z) = tj_{j_1}(z)$ with $z \in \mathsf{dom}(tj_{j_0})$.

By extension, we can also say that tj_{j_0} blocks tj_{j_1} at the *blocking point* $ptb(tj_{j_0}, tj_{j_1}) \triangleq \langle x, y\rangle$ with $y = tj_{j_1}(x)$. ◊

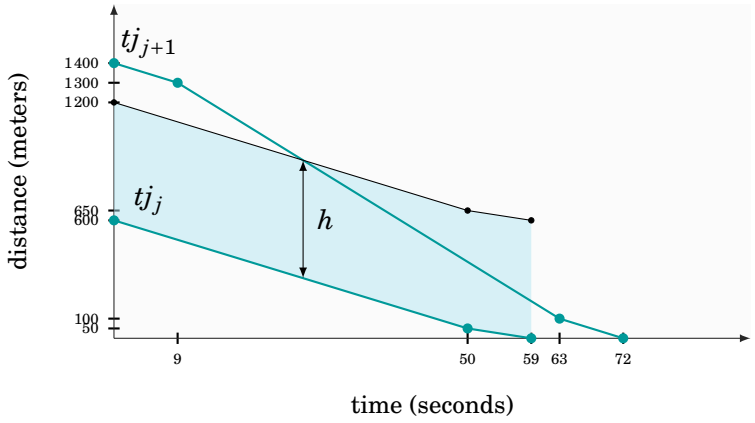


Figure 5.9: A trajectory blocking another

Figure 5.9 depicts a trajectory tj_j blocking trajectory tj_{j+1}. This configuration is undesired as it violates the safety headway h. The exclusion zone between a trajectory and its upward shift should not contain any point belonging to another trajectory when the headway h is respected.

Definition 5.6 (left shift)
Let $tj_j \triangleq \{sg_0, sg_1, \ldots, sg_{Ns_j-1}\}$ be a trajectory of size Ns_j. The *left shift* of trajectory tj_j by δ time units consists in a horizontal translation to the left hand side of the trajectory followed by a truncation, removing the part with resulting negative abscissae. It leads to the trajectory $tj_j^{[-\delta]} \triangleq \{sg_{j,k}^{[-\delta]} \mid k = 0, 1, \ldots, Ns - K - 1\}$ such that:

- $K \in \{0, 1, \ldots, Ns-1\}$ is the smallest index for which $x_{j,K,1} - \delta > 0$ (index of the first segment containing positive abscissae after a horizontal translation by δ);
- $sg_{j,0}^{[-\delta]} \triangleq [\langle 0, tj_j(\delta)\rangle, \langle x_{j,K,1} - \delta, y_{j,K,1}\rangle]$ (segment K is translated then truncated at $x = 0$ to form the first segment of the new trajectory); and
- $\forall k = 1, 2, \ldots, Ns-K-1 : sg_{j,k}^{[-\delta]} \triangleq [\langle x_{j,k+K,0} - \delta, y_{j,k+K,0}\rangle, \langle x_{j,k+K,1} - \delta, y_{j,k+K,1}\rangle]$ (remaining segments are translated to the left hand side to form the new segments). ◊

5.3.2 Trajectory Petri nets

Definition 5.7 (trajectory place)
A *trajectory place* is a place that contains trajectories instead of tokens. Each trajectory place p_i has a *length* $g_i \in \mathbb{R}_{>0}$ and a *safety headway* $h_i \in \mathbb{R}_{>0}$. ◊

When a rail system is modeled with a trajectory Petri net, trajectory places represent track portions that can be occupied by several trains at the same time. Trajectories inside trajectory places are the expected trajectories for trains traveling on these track portions at a given instant.

Definition 5.8 (place content)
We define the *content* of a trajectory place p_i as a sequence of trajectories $ct(p_i) \triangleq\triangleq \{tj_{i,j} \mid j = 0, 1, \ldots, n-1\}$ where $n \in \mathbb{N}$ is the number of trajectories of p_i. Note that a place content can be empty. ◊

Given a place content $ct(p_i)$, we write:

- $sg_{i,j,k}$ to denote the k^{th} segment of trajectory $tj_{i,j}$;
- $pt_{i,j,k,l}$ to denote the l^{th} point of segment $sg_{i,j,k}$, with $l \in \{0, 1\}$; and
- $x_{i,j,k,l}$ and $y_{i,j,k,l}$ to denote the abscissa and ordinate of point $pt_{i,j,k,l}$, respectively.

Definition 5.9 (content consistency)
We say that a given place content $ct(p_i) \triangleq \{tj_{i,j} \mid j = 0, 1, \ldots, m-1\}$ is *distance-consistent* w.r.t. $d \in \mathbb{R}_{>0}$ iff all trajectories in the place content are below distance d, i.e., $\forall j = 0, 1, \ldots, m-1 : y_{i,j,0,0} \leq d$

We say that $ct(p_i)$ is *headway-consistent* w.r.t. $h \in \mathbb{R}_{>0}$ iff no trajectory blocks another, i.e., iff:

$$\forall j = 0, 1, \ldots, m-2, \forall x \in \mathbb{R}_{\geq 0} \cup \{+\infty\} : x \in \mathsf{dom}(tj_{i,j}) \implies tj_{i,j+1}(x) \geq tj_{i,j}^h(x).$$

A place content is *consistent* iff it is distance-consistent and headway-consistent. We require, in particular, that a place content $ct(p_i)$ is distance-consistent w.r.t. g_i and headway-consistent w.r.t. h_i. ◊

We can now define trajectory Petri nets.

Definition 5.10 (trajectory Petri net)
A trajectory Petri net is a tuple $\langle P, T, M_0, A, \mathscr{F}, \mathscr{D}, \mathscr{H}\rangle$ where:

- $P \triangleq P_{Tr} \cup \mathrm{Pb}$ is a nonempty set of places, partitioned into a set of trajectory places and a set of boolean places;
- T is a nonempty set of transitions;
- M_0 is the initial marking, a function that associates an integer number of tokens to each boolean place, and a consistent place content to each trajectory place;
- $A \triangleq {}^\bullet A \cup A^\bullet$ defines the flow relation: ${}^\bullet A \subseteq P \times T$ and $A^\bullet \subseteq T \times P$;
- $\mathscr{F} : T \to \Sigma_{\mathrm{cdf}}$ assigns a CDF to each transition;
- $\mathscr{D} : P_{Tr} \to \mathbb{R}_{>0}$ assigns a distance (length) $g_i \triangleq \mathscr{D}(p_i)$ to each trajectory place p_i; and
- $\mathscr{H} : P_{Tr} \to \mathbb{R}_{>0}$ assigns a safety headway $h_i \triangleq \mathscr{H}(p_i)$ to each trajectory place p_i. ◊

We impose the structural restriction that each transition has at most, one trajectory place in its preset and at most one trajectory place in its postset. Additionally, a transition can have a trajectory place in its preset iff it has one in its postset. These are sufficient conditions for the representation of rail systems, and necessary for the chosen semantics for this model (detailed in Section 5.3.3).

The semantics of trajectory Petri nets is defined in terms of discrete and timed moves. Each event modifies the content of one or more trajectory places through the application of a combination of operations. Possible operations are:

- creation of a new trajectory in a place content;
- destruction (removal) of a trajectory from a place content;
- blocking of trajectories in a place content;
- unblocking of trajectories in a place content;
- adaptation of trajectories to comply with the safety headway; and
- application of a left shift on trajectories.

These operations are detailed below.

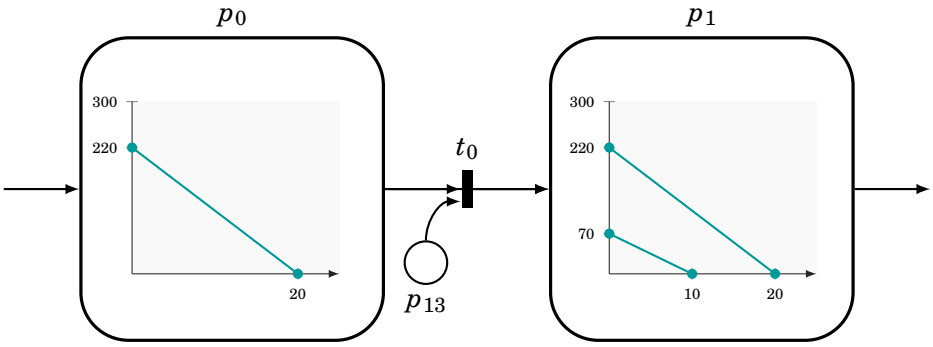


Figure 5.10: Trajectories in places of a trajectory Petri net

In what follows, we will consider a place p_i of length g_i, and its content $ct(p_i)$ with Nj trajectories. When referring to a given trajectory $tj_{i,j} \in ct(p_i)$, we will denote by $Ns_{i,j} \triangleq |tj_{i,j}|$ the number of its segments.

Creation of a trajectory

Creating a new trajectory in a place content $ct(p_i)$ consists in appending a trajectory $tj_{i,Nj}$ that covers the length g_i of place p_i. It aims at modeling the entry of a new entity (train) in the place at position $y = g_i$ and provides its desired trajectory. The form of this trajectory can vary depending on different speed profiles, and acceleration and deceler-

ation phases. We will here describe a simple version in which the created trajectory is abstracted to a single segment trajectory defined based on g_i and a given time $\tau \in \mathbb{Q}_{>0}$.

The creation process consists in generating a trajectory that starts at the beginning of the place ($pt_{i,Nj,0,0} \triangleq \langle 0, g_i \rangle$) and ends at its end ($y_{i,Nj,0,1} \triangleq 0$), with $x_{i,Nj,0,1} \triangleq \tau$, assuming τ is known. When defining the semantics of a trajectory net in terms of operations, τ will be a sampled random duration. The resulting trajectory will be $tj_{i,Nj} \triangleq [\langle 0, g_i \rangle, \langle \tau, 0 \rangle]$.

Destruction of a trajectory

The *destruction* of a trajectory $tj_{i,j}$ consists in removing it from $ct(p_i)$, and reindexing the remaining trajectories, starting from 0 while keeping the same order.

The destruction of a trajectory $tj_{i,j}$ is possible only if:

- $j = 0$ (it is the first trajectory); and
- $tj_{i,j} = \{[\langle 0, 0 \rangle, \langle 0, 0 \rangle]\}$ (it is single point-shaped).

Adaptation of a trajectory

When a trajectory $tj_{i,j}$ of size $Ns_{i,j}$ in a place content $ct(p_i)$ blocks another trajectory $tj_{i,j+1}$ of size $Ns_{i,j+1}$, an *adaptation* of trajectory $tj_{i,j+1}$ is required to ensure that it is always kept at a distance at least equal to the safety headway h_i of place p_i. This adaptation operation replaces trajectory $tj_{i,j+1}$ by a new trajectory called its *residue*.

Before defining the residue, let us use the following contextual notations:

- $ptb \triangleq \langle xb, yb \rangle$ denotes the blocking point $ptb(tj_{i,j}, tj_{i,j+1})$ (cf. Def. 5.5);
- kb_0 denotes the index of the segment in trajectory $tj_{i,j}$ that contains point ptb (i.e., $ptb \in sg_{i,j,kb_0}$), kb_1 denotes the index of the segment in trajectories $tj_{i,j+1}$ that contains point ptb (i.e., $ptb \in sg_{i,j+1,kb_1}$);
- $pth \triangleq \langle xh, yh \rangle$ denotes the point of intersection between the horizontal line of ordinate $y = h_i$ and trajectory $tj_{i,j+1}$, with $xh \triangleq \min\{x \in \mathrm{dom}(tj_{i,j+1}) \mid tj_{i,j+1}(x) = h_i\}$;
- kh denotes the index of the segment of trajectory $tj_{i,j+1}$ containing point pth; and
- Ns_0 and Ns_1 respectively denote $Ns_{i,j}$ and $Ns_{i,j+1}$.

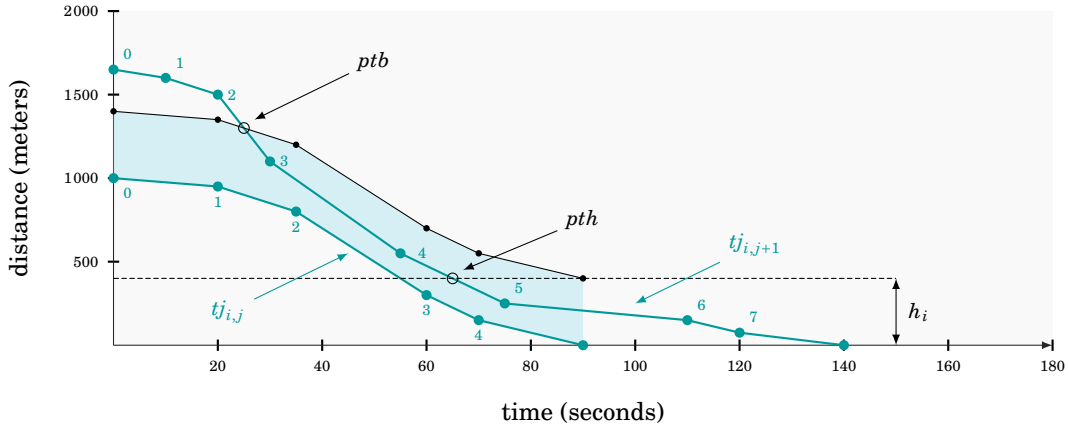


Figure 5.11: Example of a situation where an adaptation of trajectory $tj_{i,j+1}$ is needed

Fig. 5.11 shows two trajectories in a place content $ct(p_i)$ with a headway h_i. In this example, trajectory $tj_{i,j}$ has $Ns_0 = 5$ segments, and trajectory $tj_{i,j+1}$ has $Ns_1 = 8$ segments.

Trajectory $tj_{i,j}$ blocks trajectory $tj_{i,j+1}$ at point ptb; and ptb belongs to segments $kb_0 = 1$ and $kb_1 = 2$.

Definition 5.11 (residue)
Let $ct(p_i)$ be a content of a trajectory place p_i of headway h_i, and $tj_{i,j}$ a trajectory blocking another trajectory $tj_{i,j+1}$ at point ptb. The residue of trajectory $tj_{i,j+1}$ w.r.t. trajectory $tj_{i,j}$ and headway h_i is the trajectory:

$$tj_{i,j+1} \setminus tj_{i,j} \triangleq tj'_{i,j+1} \triangleq \{sg'_{i,j+1,k} \mid k = 0,1,\ldots,Ns_1 + kb_1 + Ns_0 - kb_0 - kh + 2\}.$$

Segments of the residue trajectory are constructed as follows:

(a) The first segments of trajectory $tj_{i,j+1}$ stay unchanged up to the segment containing ptb, i.e., segment $sg_{i,j+1,kb_1}$.

$$sg'_{i,j+1,k} \triangleq sg_{i,j+1,k} \qquad \text{if } k \in \{0,1,\ldots,kb_1 - 1\}.$$

Here, this part of trajectory $tj_{i,j+1}$ respects the headway h_i and can thus be kept as is.

(b) The segment $sg_{i,j+1,kb_1}$ containing ptb is replaced by two segments. The first segment starts with the first point of $sg_{i,j+1,kb_1}$ and ends with point ptb.

$$sg'_{i,j+1,k} \triangleq [pt_{i,j+1,k,0}, ptb] \qquad \text{if } k = kb_1.$$

Here we split segment $sg_{i,j+1,kb_1}$ in two parts and keep only the part of the segment that does respect the headway h_i.

The second segment start with point ptb and ends with point $pt^h_{i,j,kb_0,1}$ inherited from the shadow of trajectory $tj_{i,j}$ (w.r.t. headway h_i).

$$sg'_{i,j+1,k} \triangleq [ptb, pt^h_{i,j,kb_0,1}] \qquad \text{if } k = kb_1 + 1.$$

This new segment keeps a distance of h_i between trajectory $tj_{i,j}$ and the new trajectory $tj'_{i,j+1}$.

(c) The following segments are copies of segments of trajectory $tj^h_{i,j}$ until the last segment sg^h_{i,j,Ns_0-1}.

$$sg'_{i,j+1,k} \triangleq sg^h_{i,j,kb_0-kb_1+k-1} \qquad \text{if } k \in \{kb_1 + 2, kb_1 + 3, \ldots, kb_1 + Ns_0 - kb_0\}.$$

Here, we simply keep an exact distance of h_i between trajectories $tj_{i,j}$ and $tj'_{i,j+1}$ by copying the remaining part of $tj^h_{i,j}$ until its end.

(d) From this point on, the residue trajectory will mimic the remaining part of the original trajectory, but horizontally translated to the right hand side in order to meet at the end of segment $sg'_{i,k,kb_1+Ns_0-kb_0}$. The next segment is then composed of the last point of the shadow trajectory $tj^h_{i,j}$ and the point $pt_{i,j+1,kh,1}$ translated to the right.

$$sg'_{i,j+1,k} \triangleq [pt^h_{i,j,Ns-1,1}, pt_{i,j+1,kh,1} + pt^h_{i,j,Ns_0-1,1} - pth] \qquad \text{if } k = kb_1 + Ns_0 - kb_0 + 1;$$

with $pth \triangleq \langle tj^{-1}_{i,j+1}(h_i), h_i \rangle$.

The remaining segments are translated copies of the last segments of the original trajectory that are below $y = h_i$.

$$sg'_{i,j+1,k} \triangleq [pt_{i,j+1,k',0} + pt^h_{i,j,Ns_0-1,1} - pth, pt_{i,j+1,k',1} + pt^h_{i,j,Ns_0-1,1} - pth]$$
$$\text{if } k \in \{kb_1 + Ns_0 - kb_0 + 2, kb_1 + Ns_0 - kb_0 + 3, \ldots, kb_1 + Ns_0 - kb_0 - kh + Ns_1\};$$

with $k' \triangleq k - (kb_1 + Ns_0 - kb_0 + 2) + (kh + 1)$. ◊

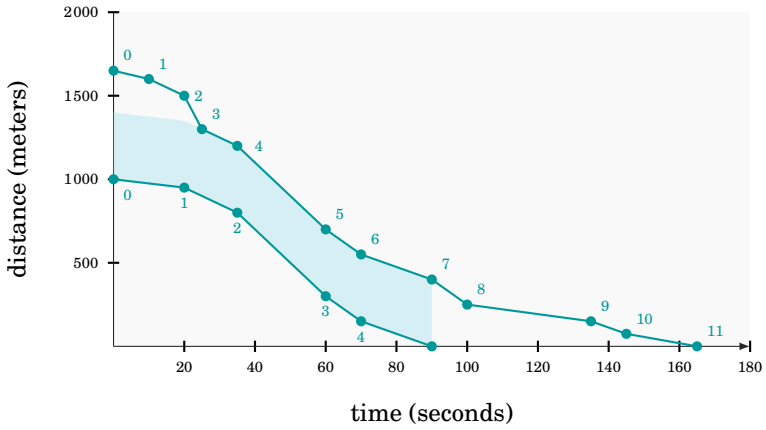


Figure 5.12: Residue

Fig. 5.12 shows the result of the adaptation of trajectory $tj_{i,j+1}$ to trajectory $tj_{i,j}$ in Figure 5.11 in order to respect the headway h_i.

Blocking a place content

Blocking a given a place content $ct(p_i)$ consists in replacing the value of the last abscissa $x_{i,0,|tj_{i,0}|,1}$ of the first trajectory $tj_{i,0}$ by $+\infty$. This operation is performed when a trajectory reaches its end (i.e. is a single point $\langle 0,0\rangle$, and the train represented by this trajectory cannot move to the next place (due to safety requirements). The first train in the place stops, and to meet safety headways, all following trains have to stop at distance h_i from their predecessor. As the duration of this stop is not known in advance, all train trajectories will end with a horizontal segment.

Blocking changes the first trajectory $tj_{i,0}$ into an horizontal ray at $y = 0$, and its upward shift into an horizontal ray at $y = h_i$ that forces adaptation of trajectory $tj_{i,1}$, i.e., replace it by its residue w.r.t. $tj_{i,0}$. This residue will end with a horizontal ray $[\langle tj^{-1}_{i,1}(h_i), h_i\rangle, \langle tj^{-1}_{i,1}(h_i), +\infty\rangle]$. Trajectory then extends to the next trajectory $tj_{i,2}$, that will end with an horizontal segment at height $2.h_i$, and so on for the whole place contents.

Unblocking a place content

When a place content is in a blocked state but safety requirements do not prevent the entity modeled by the first trajectory from reaching its next state in another trajectory place anymore, then the content should go back to a normal unblocked state (trains resume their journey. To do so, all trajectories recover their initial shape by replacing the horizontal ray part with the segments previously changed by the blocking operation.

Application of a left shift

Applying a left shift of δ time units to a place content $ct(p_i)$ simply amounts to individually applying a left shift of δ to each trajectory in $ct(p_i)$, as described in Def. 5.6. This produces a new place content $ct'(p_i)$. The main use of this transformation is to symbolize time elapsing.

5.3.3 Semantics

Now that basic operations on place contents are defiend we can give a semantics to trajectory nets. We recall briefly that a trajectory net is a variant of STPN, i.e. a tuple $\langle P, T, M_0, A, \mathscr{F}, \mathscr{D}, \mathscr{H} \rangle$ where $P, T, A, \mathscr{F}, \mathscr{D}, \mathscr{H}$ have the same meaning as in STPNs, but where subset $P_{Tr} \subseteq P$ of places contain trajectories instead of tokens. Markings hence associate place contents (sequences of trajectories) to places in P_{Tr} and standard tokens to other places.

Definition 5.12 (configuration)
A *configuration* of a trajectory Petri net is a pair $\langle M, U \rangle$ with:

- M is a marking that assigns a nonnegative integer (number of tokens) to each boolean place and a consistent place content to each trajectory place; and
- U is a function that assigns a consistent place content to some trajectory places that are blocked. These place contents contain the desired trajectories that trains should follow when they are not blocked anymore.

We denote the *unblocked content* of each trajectory place p_i by:

$$U(p_i) \triangleq \{uc_i \mid i = 0, 1, \ldots, m_i - 1\};$$

where m_i designates the number of unblocked versions of trajectories of place p_i. ◊

Definition 5.13 (usable place)
A place p_i is *usable* if the first trajectory in its content is the single point-shaped trajectory $[\langle 0,0 \rangle, \langle 0,0 \rangle]$ ◊

Definition 5.14 (enabledness)
A transition t is enabled by a marking M iff there exists at least one token in each boolean place of its preset, the trajectory place in its preset is usable.

We formally define the set of enabled transitions by a marking M by:

$$\mathsf{enab}(\mathscr{C}) \triangleq \{t \in T \mid (\forall p \in {}^\bullet t \cap \mathrm{Pb} : M(p) \geq 1) \wedge (\forall p_i \in {}^\bullet t \cap P_{Tr} : |M(p_i)| \geq 1 \wedge tj_{i,0} = \{[\langle 0,0 \rangle, \langle 0,0 \rangle]\})\}.$$

◊

Informally, as trajectory places represent track segments, a transition that has a trajectory place p_i in its preset is enabled if the first train represented by a trajectory in the contents of p_i has reached the end of the track segment.

Definition 5.15 (firability)
A transition is *firable* in a configuration $\mathscr{C}$ iff it is enabled by the marking of $\mathscr{C}$ and each place in its postset contains enough space at its beginning to insert a new trajectory.

More precisely, we define the set of firable transition in configuration $\mathscr{C}$ by:

$$\text{fira}(\mathscr{C}) \triangleq \{t \in \text{enab}(M) \mid \forall p_i \in t^\bullet \cap P_{Tr} : ct(p_i) = \varnothing \vee g_i - y_{i,|ct(p_i)|-1,0,0} \geq h_i\}. \qquad \Diamond$$

Intuitively, a transition is firable if the train movement that it represents (entering the next track portion, leaving a station, entering a station) is allowed by the safety requirements.

The execution of a trajectory Petri net starts from an initial configuration, and is defined with different types of *moves*.

Blocking move Trajectories are used to represent trains moving on a track. When a train's trajectory is the first trajectory $tj_{i,0}$ of a place p_i and is of the form $[\langle 0,0\rangle, \langle 0,0\rangle]$, it means that the train has ended it's dwell or running operation and is ready to begin its next dwell or running operation. Normally, a firable transition t in the postset of p_i is fired, $tj_{i,0}$ is removed from $M(p_i)$ and a new trajectory is created in the trajectory place $p_{i'}$ in the postset of t. However, if there is no available space in $p_{i'}$ (i.e. $g'_i - tj_k(0)$, where k is the last trajectory in p'_i is smaller than the safety headway h'_i), then no trajectory can be added to its content. In this case, the content of place p_i must be *blocked*, according to the blocking operation defined above.

A blocking move from a configuration $\langle M, U\rangle$ occurs when a place p_i contains a trajectory of the form $[\langle 0,0\rangle, \langle 0,0\rangle]$, but none of the transitions in $p_i{}^\bullet$ is firable. This blocking move leads to a new configuration $\langle M', U'\rangle$ in which:

- the marking of boolean places stays unchanged, i.e., $\forall p \in \text{Pb} : M'(p) \triangleq M(p)$.
- the content of a trajectory places p_i is updated by a blocking operation.
- copies of original trajectories before the blocking procedure are memorized by unblocked content function U'.

  $$U'(p_i) \triangleq \{tj_{i,j} \mid j = 1, 2, \ldots, Nj - 1\}.$$

  Note that there is no need to store the first trajectory $tj_{i,0}$.

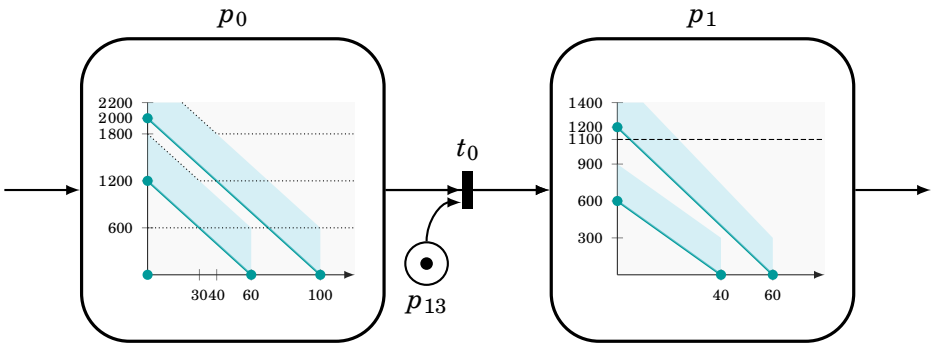


Figure 5.13: Nonfirable transition due to safety requirements

The example of Figure 5.13 shows a transition that is ready to fire but cannot due to safety requirements in place p_1. If we set $h_1 = 300$ and $g_1 = 1400$, then $tj_{1,1}(0)$ must be smaller than 1100 to let enough space in place p_1 for insertion of a new trajectory. As a consequence, the content of place p_0 is blocked, resulting in the trajectories shown in Figure 5.14.

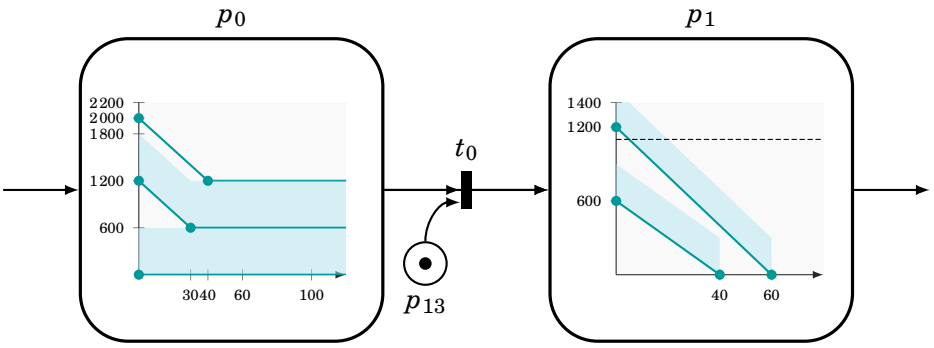


Figure 5.14: Trajectories in a blocked place

A blocking move is denoted by: $\langle M,U\rangle \xrightarrow{\text{block}(p_i)} \langle M',U'\rangle$. We now give the semantics rule for blocking

$$\frac{U(p_i) = \epsilon \wedge p_i \text{ is usable}}{\langle M,U\rangle \xrightarrow{\text{block}(p_i)} \langle M',U'\rangle}$$

Timed move

A timed move from a configuration $\langle M,U\rangle$ consists in simply letting time elapse for a duration δ when no *action* (a discrete move or a blocking move) is possible. The value of δ corresponds to the maximum amount of time that can elapse and after which an action becomes possible. A time move is performed in one step through application of a left shift of δ to contents of all trajectory places.

Let us assume that in configuration $\langle M,U\rangle$. For each place, we can define the following quantities.

$d_\alpha(p_i)$ is the time needed for the first trajectory in $ct(p_i)$ to becomes a point, i.e. letting $tj_{i,0}$ be this trajectory, $d_\alpha = tj_{i,0}^{-1}(0)$;

$d_\beta(p_i)$ is the time needed before the last trajectory in the contents of p_i leaves a sufficient headway for insertion, i.e. if $tj_{i,k}$ is the last trajectory in the place content of p_i, $d_\beta(p_i) = tj_{i,k}^{-1}(g_i - h_i)$.

Then, the time that can elapse from configuration $\langle M,U\rangle$ is a value smaller than the minimal amount of time before a place contents becomes blocked, or before a transition becomes firable, either because a place contents can be unlocked or because a train arrives at the end of its track.

$$\text{step}(\langle M,U\rangle) = \min\left(\begin{array}{l}\min\{d_\alpha(p_i) \mid U(p_i) = \emptyset\},\\ \min\{d_\beta(p_i) \mid \exists t, p_i \in t^\bullet, \exists p_j \in {}^\bullet t, U(p_j) \neq \emptyset\}\end{array}\right)$$

A timed move then consists in choosing a value $\delta \leq \text{step}(\mathscr{C})$, and then letting trajectories evolve during δ time units, i.e. perform a left shift by δ in all place contents and blocked contents: For every $p_i \in P_{Tr}$, $M'(p_i) = \text{shift}(ct(p_i),\delta)$. If p_i is blocked, then every trajectory $tj_{i,k}$ in $U(p_i)$ is shifted by a value that is the minimal value between δ and $tj_{i,k}(k.h_i)$ (U memorizes how to resume a trajectory, and hence as trajectory k becomes horizontal when reaching distance $k.h_i$, the whole part of $tj_{i,k}$ depicting the part of the trajectory from distance $k.h_i$ to 0 has to be remembered.

A timed move is denoted by: $\langle M,U\rangle \xrightarrow{\delta} \langle M',U'\rangle$. We hence have the following rule:

$$\frac{fira(\langle M,U\rangle) = \emptyset \wedge \delta \leq \mathsf{step}(\langle M,U\rangle)}{\langle M,U\rangle \xrightarrow{\delta} \langle M',U'\rangle}$$

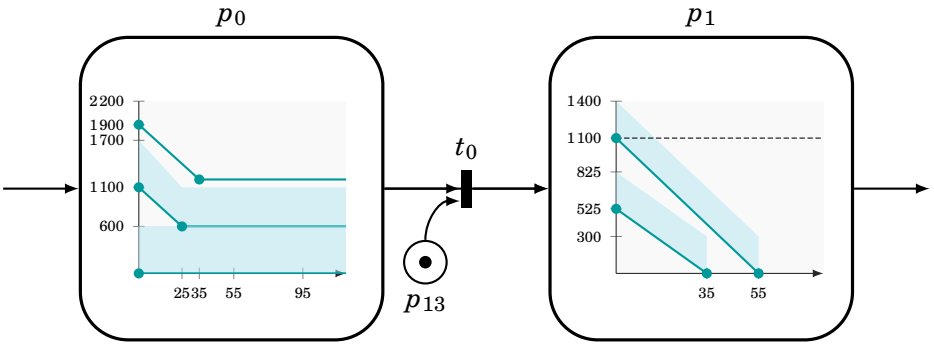


Figure 5.15: Shifted place contents

Fig. 5.15 shows the result after shifting the contents of places p_0 and p_1 from Fig. 5.14 by 5 time units. One can notice on this figure that the last trajectory of place p_1 now leaves enough space for insertion of a trajectory. As a consequence, transition t_0 becomes firable.

Discrete move

A discrete move consists in firing a transition t among the set of *firable* transitions. It moves tokens and trajectories from their current places in the preset of t to new places in the postset of t. This is performed by:

(a) removing one token from each boolean place in the preset of t:

$$\forall p \in {}^\bullet t \cap \mathrm{Pb} : M'(p) \triangleq M(p) - 1.$$

(b) removing the first trajectory from the trajectory place p_j in the preset of t:

$$M'(p_j) \triangleq M(p_j) \setminus \{tj_{j,0}\}.$$

If p_j was blocked (i.e., $U(p_j) \neq \emptyset$) replace $M(p_j)$ by $U(p_j)$ and set $U(p) = \emptyset$.

(c) adding a token to each boolean place in the postset of transition t:

$$\forall p \in t^\bullet \cap \mathrm{Pb} : M'(p) \triangleq M(p) + 1.$$

(d) creating a new trajectory $\{[\langle 0, g_i\rangle, \langle \tau, 0\rangle]\}$ in the content of the trajectory place in the postset of transition t; with τ sampled from $F_t \triangleq \mathscr{F}(t)$, and adapting it w.r.t. the last trajectory in $t^\bullet \cap P_{Tr}$, if necessary.

A discrete move is denoted by $\langle M,U\rangle \xrightarrow{t} \langle M',U\rangle$. Note that due to random sampling, there can be several successor configuration for $\langle M,U\rangle$. We denote by $Succ(\langle M,U\rangle,t)$ the set of successor configuration that can be reached when firing t from $\langle M,U\rangle$.

$$\frac{t \in fira(\langle M,U\rangle) \quad \langle M',U'\rangle \in Succ(\langle M,U\rangle,t)}{\langle M,U\rangle \xrightarrow{t} \langle M',U'\rangle}$$

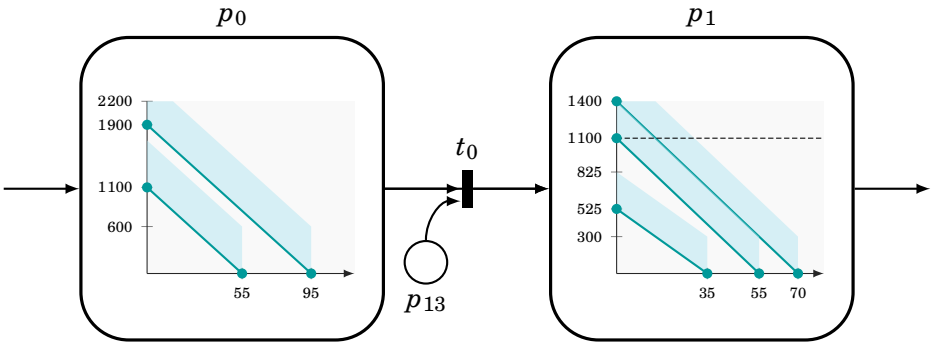


Figure 5.16: Transition firing in a trajectory Petri net

Figure 5.16 represents the configuration obtained after firing transition t_0 from the configuration in Figure 5.15. This configuration was obtained by removing the first trajectory from p_0 and creating a new trajectory in place p_1. Notice that in this example, the new trajectory added to p_1 needs not be adapted w.r.t the contents of p_1 as is does not cross the upward shift of other trajectories.

5.3.4 Comments

The trajectory nets presented in this section can model urban train systems with a moving block policy, regulation schemes, and timetables. The way to obtain a trajectory net from a given topology is the same as for blocking STPNs. The model contains several simplifications such as the linear shapes of trajectories, headways defined as fixed distances, headways associated to places instead of actual running entities (trains). Obviously, this model can be refined to consider more complex trajectories with acceleration and decelerations encoded as polynoms, dynamic headways that depend on trains speed, etc.. However, refinement of the model increases the time needed to create, block and adapt trajectories, and hence can slow down the simulation process. For example, finding the intersection between a trajectory and a trajectory shadow in the current model is easy, but using polynomial trajectories may significantly increase computation time.

This model can be seen as a discretization of a time-space diagram (TSD) in which we only have a partial and local vision of the expected trajectories of trains, and the rest is generated during the execution of the system. Note also that when a train gets close to the end of the place it is occupying (usually when it is about to enter a station), it may get closer to the preceding train in a following place than allowed in real situations. The main reason it that our model does not impose safety headway between different places. The semantics has to be refined to take into account this particular situation, and adapt the first trajectory of a given place according to the last trajectory of its following place. This can be made possible by introducing an *inter-place headway* $h_{p,p'}$ between pairs of consecutive places p and p' (in terms of topology) but this will bring additional complexity to the model, and thus longer computation times.

The model depicted in this section has been implemented, and showed good performances that are comparable to simulation times measured for STPNs with blocking semantics.

6

Realizability of schedules

In the predecing chapters, we have introduced timetables, that are a priori schedules of train operations, designed to optimize performance of urban train systems. Timetables can be seen as partial orders among basic tasks, that abstract low-level implementation details, and are decorated with dates and timing constraints. We have introduced two models, namely STPNs with a blocking semantics and trajectory nets as a model of trains behaviors of the network. If a system realizes correctly a timetable, then it meets the KPI objectives that led to the construction of this particular timetable. If incidents occur (and such incidents are unavoidable in a normal operation day) regulation algorithms have to compensate them to return to the desired timetable.

In this setting, an implicit assumption is that the timetable can be realized up to some minor corrections brought by the regulation. However, designing a correct and optimal schedule for a system is a complex problem. On one hand, occurrence dates of events can be seen as variables, and correct and optimal schedules as optimal solutions (w.r.t. some criteria) for a set of constraints over these variables. Linear programming solutions have been proposed to optimize scheduling in train networks [18, 20]. On the other hand, optimal solutions (for instance, w.r.t. completion date) are not necessarily the most probable nor the most robust ones.

Consequently, optimal and realizable schedules are not necessarily robust enough if they impose tight realization dates to systems that are subject to random variations. Further, schedules and models for physical networks need not be built simultaneously, and some discrepancies can appear between the desired schedule, and the behaviors that can be implemented by the low-level system. Hence, nothing guarantees a priori that the system is able to *realize* a given schedule.

This calls for tools to check ***realizability*** of a schedule. One can expect systems designers and timetable experts to share a common understanding of the behavior of their system; so, in general, the answer to the boolean realizability question *"Is the given schedule realizable by the system?"* should be positive. However, being able to realize a schedule does not mean that the probability to comply with a particular schedule is high enough. Obviously, in systems with random delays which values are sampled from continuous

distributions, the probability to realize a schedule with precise dates is null. Beyond boolean realizability, a schedule shall, thus, be considered as *realizable* if it can be realized up to some bounded imprecision, and with a significant probability. This leads to the notion of probabilistic realizability.

In this chapter, we address realizability of schedules by STPNs with a blocking semantics introduced in section 5.2. We consider scheduling of important events identified in a timetable, and not necessarily the whole table, which leaves more flexibility to realize a particular schedule. We use a partial order semantics, unfoldings [50, 26], symbolic techniques [14, 16] and transient analysis [39] to address the questions of boolean and probabilistic realizability.

The chapter is organized into 5 parts. We first give a precise definition of schedules and of the realizability questions We then propose a notion of process for STPNS with a blocking semantics. Surprisingly, extensions of unfoldings and processes is not a straightforward, as blocking introduces timing constraints that have to be considered and changes the standard notion of urgency. We introduce symbolic representation of time constraints in unfoldings to represent possible timed executions of a blocking STPN. With this model, one can address infinite classes of processes symbolically. However, this raises an important issue, as satisfiability of a set of constraints on occurrence dates of events is not monotonous. Fortunately, unfolding a net up to a sufficient depth allows to characterize processes that can be executed in a fixed amount of time.

This central result is then used to address boolean realizability: the problem is solved by checking that a particular schedule can be embedded into a process of a bounded unfolding of a net. Once the (finite) set of processes that embed a schedule are characterized, we use it to evaluate the probability to realize at least one of them, leading to the notion of probabilistic realizability. We last show how to check boolean and quantitative realizability on a practical example; that is, verification of correct scheduling of train trips in an RTS.

The results presented in this chapter were published in [34, 35], we hence refer interested readers to these papers for proofs of theorems introduced in this chapter.

6.1 Schedules

Schedules are objects of everyday's life, and are both used as documentation for a system (for instance, bus and train schedules indicate to users when and where to take a metro), and to guide a system along behaviors that guarantee some quality of service. In the context of transportation systems, schedules are known as *timetables*.

As shown in Chapter 4 the natural notion to encode timetables is a partially ordered sets of events decorated with dates. Events represent the beginning or end of a task, the departure or arrival of a train or a bus, etc. Partial ordering among events allows to account for linearity in individual trajectories, and for causal dependencies due to exclusive resource use. The dates decorating these partial orders are dates that comply with the dependencies, and also with some physical characteristics of the modeled systems: a train move from a station st_i to the following one st_{i+1} in its itinerary takes time, which shall be reflected by the dates attached to departures and arrivals at different stations.

Now, one can be interested in higher-level views of the desired behavior of a system, and abstract away some events and dependencies that appear in a full timetable. This is captured by the notion of schedule, which is a way to describe an ordering on a subset of events occurring in a system, and can be interpreted as some form of partial requirement. This raises the question of whether a particular schedule can be effectively realized by the system. As schedules and system models are descriptions of the same system, one can expect the answer to be positive in most of cases. However, solutions returned by a solver for a constraint problem are optimal solutions w.r.t. some criteria, but not necessarily the most plausible nor the most robust ones. Indeed, for obvious reasons, one cannot ask a bus to reach each stop at the earliest possible date: such solution makes systems poorly robust to random delays, that necessarily occur in transport systems. Providing the ability to check that a schedule is realizable with reasonable chances is, hence, an essential tool to design schedules. A way to answer this question is to ask the probability that a schedule is realized by the system.

Formally, a schedule describes causal dependencies among tasks, and timing constraints on their respective starting dates. Schedules are defined as decorated partial orders, and allow timing constraints among tasks that are not causally related.

Definition 6.1 (schedule)
Formally speaking, a *schedule* over a finite alphabet $\mathscr{A}$ is a quadruple $S \triangleq \langle N, \rightarrow, \lambda, C \rangle$ where

- N is a set of nodes,
- $\rightarrow \subseteq N \times N$ is an acyclic precedence relation,
- $\lambda : N \rightarrow \mathscr{A}$ is a labeling of nodes, and
- $C : N \times N \nrightarrow \mathbb{Q}_{>0}$ is a partial function that associates a minimum time constraint to pairs of nodes.

A *dating function* for a schedule S is a function $d : N \rightarrow \mathbb{Q}_{\geq 0}$ that satisfies all constraints of C and $\rightarrow$, i.e., $\langle n, n' \rangle \in \rightarrow \Rightarrow d(n) \leq d(n')$, and $x = C(n, n') \Rightarrow x \leq d(n') - d(n)$. $\Diamond$

This model for schedules borrows many ingredients from timetables, but with the ability to impose timing constraint on events that are not causally related. Intuitively, if $x = C(n, n')$, then n' cannot occur earlier than x time units after n, and if $\langle n, n' \rangle \in \rightarrow$, then n precedes n' (causally). Constraints model, for example, the minimal times needed to perform tasks and initiate the next ones in production cells, or the times needed for trains to move from a station to another. A schedule S is *consistent* if the graph $\langle N, \rightarrow \mathrm{def}(C) \rangle$ does not contain cycles ($\mathrm{def}(C)$ is the domain of definition of the partial function C). Naturally, consistent schedules admit at least one dating function.

Even consistent schedules might not be realizable by an existing physical system, due for instance to trip durations. When a schedule is realizable, the probability to achieve it can also be very low.

Consider the example of Figure 6.1. This Figure represents the beginning of a schedule for three trains, where only departure dates are planned. The departures for each train are depicted as boxes, carrying a label of the form $i : dJ$, where i is the event number, and dJ means that this event is a departure from station J. Furthermore, an execution date is attached to each node. One can see on this drawing that schedules are partial orders containing train trajectories. However, there are some dependencies among events from distinct trains: for instance, our schedule imposes that the second departure of the day

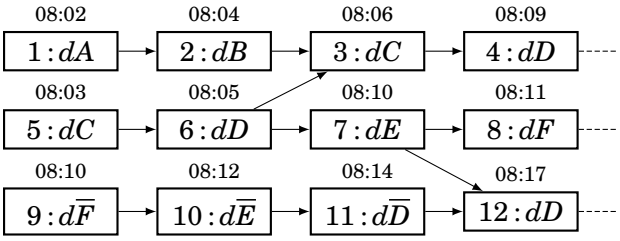


Figure 6.1: A possible schedule for train departures in a metro network

at station D (event $6 : dD$) precedes departure $3 : dC$ from station C. Similarly, $7 : dE$ must precede $12 : dD$. This schedule makes sense, as some metro networks are operated only according to the departure dates and the end of a line. Now, a sensible question is: *"Can a network realize this schedule?"*; or, more precisely, *"Can the STPN designed for this network (with additional control places) realize such a schedule starting at 08:00 with a train at station A, a train at station C, and a train at station $\overline{F}$?"* The answer to this question is not straightforward. Even if the answer is positive, the next step is to ask whether this schedule, with a tolerance of, e.g., 1 minute delay for each departure, is probable enough. If the answer is that the probability to realize our schedule is very low, then this schedule should not be considered as operational.

6.2 Partial order semantics of STPNs

Let us consider the representation of a piece of network shown in Figure 6.2. In this example, a train is dwelling at a station A (represented by p_0). When its dwelling time is up after a certain duration, and it is ready to depart, the train can be controlled to move towards two different destinations B and C. This translates into, either the firing of transition t_0, or t_1, to respectively transfer the token into place p_1, or p_2. Similarly, if the token is in place p_2 it will reach p_3 through the firing of t_2. However, decision of firing t_0 or t_1 are dependent on the content of additional control places p_4 and p_5. Indeed, firing t_0 requires a token in p_0 and p_4 while firing t_1 requires tokens in p_0 and p_5. Once can also notice that control places p_4 and p_5 share one token between them at all times. This simple mechanism allows to model guiding of trains towards different possible routes. PDFs f_0, f_1, and f_2 respectively attached to transitions t_0, t_1, and t_2 are represented at the bottom of Figure 6.2.

In this example, a train staying at station A will leave after a dwell time comprised between 60 and 80 seconds, provided that its destination place is empty. Dwell times in place p_0 have different distributions in the interval $[60, 80]$, depicted by functions f_0 and f_1. Distribution choice depends on whether the train leaves for station B or station C. The run from A to C lasts between 130 and 140 seconds, and the distribution of running time is depicted by function f_2. Now, if a train is ready to leave station A to go to C after a sufficient dwell time, but another train already occupies the track portion from A to C, then departure is forbidden. In our model, this is implemented by the elementary semantics that says that a transition can fire only when its postset is empty. In particular, in our example, this means that a token will enter place p_1 (resp. place p_2) at the earliest 60 seconds after p_0 was filled, and at the latest 80 seconds after, if p_4 (resp. p_5) contains a token. However, even if p_4 (resp. p_5) contains a token, there is no guarantee that the

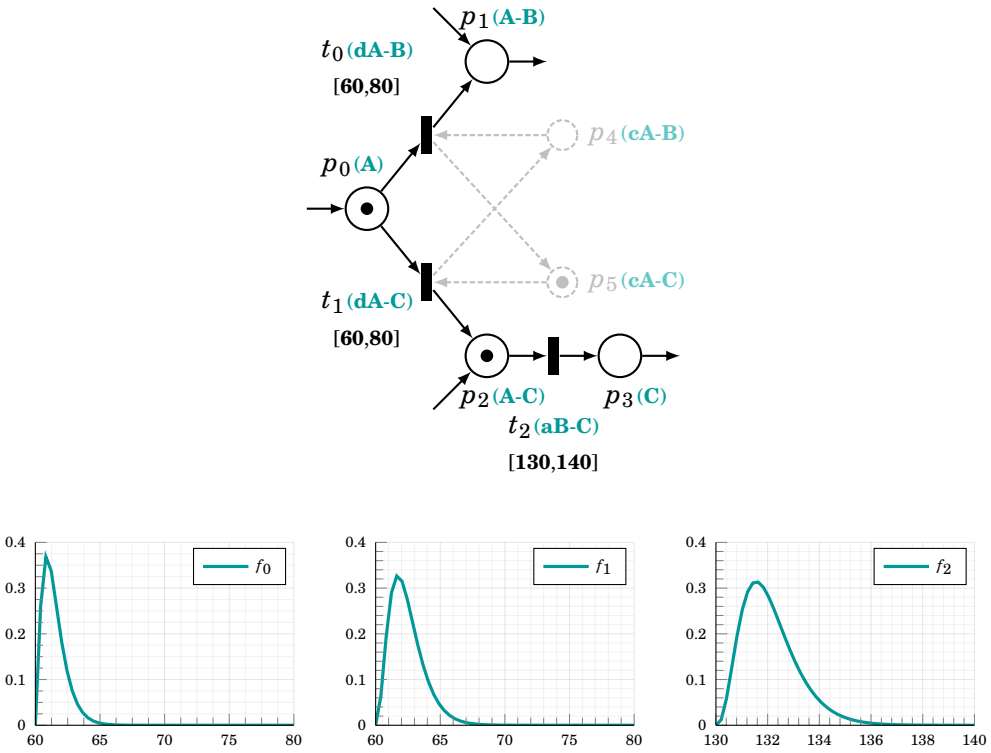


Figure 6.2: STPN representation of a shunting mechanism in a rail network

dwell time of a token in p_0 is smaller than 80 seconds, despite urgency. Indeed, if places p_5 and p_2 are filled, transition t_1 has to wait for p_2 to be empty to fire, which may occur at the earliest 130 seconds and at the latest 140 seconds later.

One of the possible executions of the STPN of Figure 6.2 is the following: Assuming that the value sampled for t_1 and t_2 are respectively $\zeta_1 = 62$ and $\zeta_2 = 132$, then time ζ_1 is elapsed after 62 time units, but t_1 cannot yet fire as place p_2 is occupied. Hence, one has to wait 132 time units, fire t_2, and then fire t_1.

In terms of *timed word*, this execution can be represented as $u \triangleq \langle t_2, 132\rangle \cdot \langle t_1, 132\rangle$. Equivalently, one can describe this execution as a *time process* TP of $\mathcal{N}_0$ (see later in this section for the formal definition). Roughly speaking, a time process *unfolds* the STPN and associates an execution date to occurrences of transitions. The time process corresponding to timed word u is given in Figure 6.3. Note, on this example, that even if the first occurrences of t_1 and t_2 seem to be concurrent (they occur at the same dates, and are not causally related), the elementary semantics imposes that t_2^0 occurs before t_1^0.

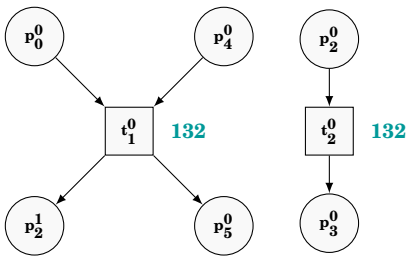


Figure 6.3: A time process of $\mathcal{N}_0$

As already highlighted in [6] for TPNs, timed words give a sequential and interleaved view for executions of inherently concurrent models. Time processes, on the other hand, bring a noninterleaved semantics. They are defined as *causal nets* equipped with dating functions.

Definition 6.2 (causal net)
A causal net is a finite acyclic net $\mathrm{Cn} \triangleq \langle P, T, A\rangle$ where $\forall p \in P : |p^\bullet| \leq 1 \wedge |{}^\bullet p| \leq 1$; i.e., each place in a causal net has, at most, one transition in its preset (no *merging*) and, at most, one transition in its postset (no *conflict*). ◊

Definition 6.3 (causality)
Given a pair of places $p, p' \in P$, we define the causality relation $\prec$ and write $p \prec p'$ iff $p^\bullet \cap {}^\bullet p' \neq \varnothing$. We denote by $\preceq$ the reflexive transitive closure of $\prec$. When places p and p' are incomparable with $\preceq$, we say that they are *concurrent*, and write $p \parallel p'$ iff $p \not\preceq p'$ and $p' \not\preceq p$. Theses definitions extend to transitions. ◊

Definition 6.4 (time process)
A time process is a pair $\mathrm{TP} \triangleq \langle \mathrm{Cn}, \theta\rangle$ where

- $\mathrm{Cn} \triangleq \langle B, E, A\rangle$ is a causal net, and
- $\theta : E \to \mathbb{R}_{\geq 0}$ assigns to each transition in E a nonnegative real date, that is coherent with causality: $\forall e, e' \in E : e \preceq e' \implies \theta(e) \leq \theta(e')$.

In time processes, places in B are called *conditions*, and transitions in E are called *events*. The *depth* of a time process TP is the total number of events found in the longest path of the graph $\langle B \cup E, A\rangle$. ◊

Intuitively, in a time process TP of an STPN $\mathcal{N}$, events in E represent occurences of firings of transitions of $\mathcal{N}$, and conditions in B represent the conditions needed to fire these transitions (i.e. occurrences of place fillings) and the result of transitions firings.

We denote by $\mathsf{tr}(e)$ the transition t attached to an event e, and by $\mathsf{pl}(b)$ the place p associated with a condition b. To differentiate between occurrences of transition firings, an event will be defined as a pair $e \triangleq \langle X, t\rangle$, where t is the transition whose firing is represented by e and X is the set of conditions it requires. Similarly, a condition is defined as a pair $b \triangleq \langle e, p\rangle$, where e is the event whose occurrence generates condition b, and p is the place whose filling is represented by b. The flow relations are, hence, implicit: ${}^\bullet e \triangleq \{b \in X \mid e = \langle X, t\rangle\}$ and similarly $e^\bullet \triangleq \{b \in B \mid b = \langle e, p\rangle\}$, and for $b \triangleq \langle e, p\rangle$, ${}^\bullet b \triangleq e$ and $b^\bullet \triangleq \{e \in E \mid b \in {}^\bullet e\}$. (n.b., as the preset ${}^\bullet b$ of a condition b is always a singleton, we slightly abuse notations and write $e = {}^\bullet b$ instead of the set $\{e\} = {}^\bullet b$.) Following these definitions, we will often drop flow relations and simply refer to time processes as triples of the form $\mathrm{TP} \triangleq \langle B, E, \theta\rangle$.

Given an STPN $\mathcal{N}$, for every timed word u in the timed language of $\mathcal{N}$, we can compute a time process TP_u depicting execution of events occurring in u. The construction of a time processes from a timed word described below is the same as in [6]. It does not consider probabilities. As the construction starts from an executable word of $\mathcal{N}$, the construction does not need to address blocking. We denote by TP_u the time process obtained from a timed word $u = \langle a_0, dt_0\rangle\langle a_1, dt_1\rangle \ldots \langle a_{q-1}, dt_{q-1}\rangle \in \mathscr{L}(\mathcal{N})$. It can be built incrementally by adding transition occurrences with their associated dates, one after the other, starting from an initial set of conditions $B_0 \triangleq \{\langle p, \bot\rangle \mid p \in M_0\}$ where $\bot$ is a dummy event used as a generator of initial conditions. This event is not represented in the graphical illustrations

of time processes. Intuitively, this construction adds one event after another, following the ordering and dates given in u. At each step, newly appended events are attached to maximal conditions in the already built process.

Definition 6.5 (maximal conditions)
A condition $b \in B$ in a time process $\mathrm{TP} = \langle B, E, \theta \rangle$ is *maximal* if it has an empty postset. We denote by $\mathsf{maxb}(\mathrm{TP})$ the set of maximal conditions of a time process TP, i.e.,

$$\mathsf{maxb}(\mathrm{TP}) \triangleq \{b \in B \mid b^\bullet = \varnothing\}.$$

$\lozenge$

We can now give an inductive construction technique to build a time process TP_u from a timed word $u = \langle t_1, dt_1 \rangle \cdot \langle t_2, d_2 \rangle \cdots \langle t_q, dt_q \rangle \in \mathscr{L}(\mathscr{N})$:

At each step $i \leq q$, $\mathrm{TP}_{u,i} \triangleq \langle B_i, E_i, \theta_i \rangle$ denotes the time process built after i steps, i.e. after reading the the prefix
$\langle a_1, dt_1 \rangle \ldots \langle a_i, d_i \rangle$ of u.

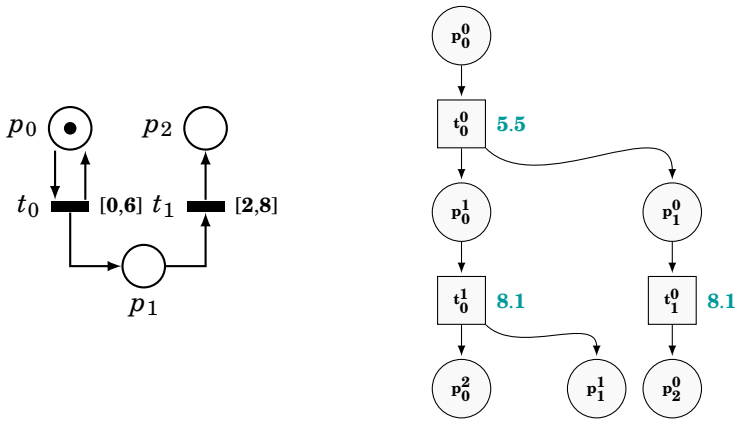
The construction starts at step $i = 0$ from the initial time process $\mathrm{TP}_{u,0} \triangleq \langle B_0, E_0, \theta_0 \rangle$. We use a dummy event $\perp$ that sets the initial contents of places according to M_0. We then have: $B_0 \triangleq \{\langle \perp, p \rangle \mid p \in M_0\}$, $E_0 \triangleq \{\perp\}$, $\theta_0 : \{\perp\} \to \{0\}$, as the execution starts at date 0.

Then, at every step $i \in \{1, \ldots, q\}$, we build $\mathrm{TP}_{u,i} = \langle B_i, E_i, \theta_i \rangle$ from $\mathrm{TP}_{u,i} = \langle B_{i-1}, E_{i-1}, \theta_{i-1} \rangle$ as follows:

We compute the set of maximal conditions that are occurrences of places in the preset of t_i. Formally, this set is $X_i \triangleq \{b \in \mathsf{maxb}(\mathrm{TP}_{u,i-1}) \mid \mathsf{pl}(b) \in {}^\bullet t_i\}$. The new event added is $e_i \triangleq \langle t_i, X_i \rangle$. Then,

- $B_i \triangleq B_{i-1} \cup \{\langle p, e_i \rangle \mid p \in t_i^\bullet\}$;
- $E_i \triangleq E_{i-1} \cup \{e_i\}$
- $\theta_i(e) = \theta_{i-1}(e)$ if $e \in E_{i-1}$ and $\theta_i(e) = d_i$ if $e = e_i$.

The construction ends with $\mathrm{TP}_u \triangleq \mathrm{TP}_{u,q}$.



(a) (b)

Figure 6.4: An example STPN $\mathscr{N}_1$ and one of its time processes

Figure 6.4-b is an example of a time process of STPN $\mathcal{N}_1$. In this example, event t_i^j and condition p_i^j respectively denote the $(j-1)^{\text{th}}$ occurrence of transition t_i and place p_i. This time process corresponds to the timed word $u = \langle t_0, 5.5\rangle \cdot \langle t_1, 8.1\rangle \cdot \langle t_0, 8.1\rangle \in \mathscr{L}(\mathcal{N}_1)$. It contains causal dependencies among events (e.g., from t_0^0 to t_1^0). One can also notice that, due to the blocking semantics, event t_0^1 cannot occur before t_1^0 as t_0 cannot fire as long as place p_1 is filled. However, this information is not explicitly shown in the time process description. The timed language $\mathscr{L}(\mathcal{N})$ of a TPN can be reconstructed as the set of linearizations of its time processes.

Given a time process TP of $\mathcal{N}$, one can find a word u such that $\text{TP}_u = \text{TP}$. Such a word is obtained by finding a total order on events in TP that considers causality and dates of events, but also blocking. Once a total ordering $e_1 ... e_{|\text{TP}|}$ where $e_i = \langle t_i, X_i\rangle$ is found $u = (t_1, \theta(e_1)) \dots (t_{|\text{TP}|}, \theta(e_{|\text{TP}|}))$. Preserving causality and time means that for a pair of events e, e' with $e \neq e'$, e must appear before e' in the generated word if $\theta(e) < \theta(e')$ or if $e \preceq e'$. Note that due to blocking semantics, some causality and time-preserving interleavings may not produce valid timed words of $\mathscr{L}(\mathcal{N})$: in the time process of Figure 6.4-b, t_0^1 cannot occur before t_0^1, even if both transitions have the same date. A correct ordering among events with identical dates in a process TP_u can, however, be found by checking that a chosen ordering does not prevent occurrence of other transitions.

Realizability

Elementary semantics allows to handle safety requirements for systems with critical resources. However, this semantics makes reasoning on systems harder, and brings no guarantee on the liveness of a system. On our example of Figure 6.2, a possible wish of designers is that trains stay no longer than 80 seconds at station A, and one train out of two goes to C. The second requirement is easily implemented by the simple additional control represented by the dashed places and flows. However, nothing guarantees a priori the first requirement, i.e., that place p_2 is emptied at the latest 80 seconds after a token has entered p_0. So, even if the model seems correct, trains may have to wait longer than expected a priori.

The question of whether a STPN can implement a predetermined schedule is called *realizability*. To check whether a consistent schedule S can be realized by a system, described as an STPN $\mathcal{N}$. we want to verify that there exists a time process TP of $\mathcal{N}$ and an embedding function ψ mapping abstract events of S onto concrete events of TP, such that causally related events of S are causally dependent in TP, and dates of events in TP meet the constraints on their abstract representation.

In Sections 6.5 and 6.6, we show how to check realizability of a schedule S by an STPN $\mathcal{N}$, and how to compute a lower bound on the probability that S is realized by $\mathcal{N}$. This allows in particular to check that the probability of realization of a schedule is strictly positive, or above a given threshold.

6.3 Unfolding of STPNs with blocking semantics

A time process emphasizes concurrency but only gives a partial order view of a *single* timed word. However, a net may have an infinite number of time process. Further, an infinite number of time processes can have the same structure but different dating functions. This is for instance the case of the process in Figure 6.4-b. It is, hence, interesting

to consider *symbolic time processes*, that define constraints on events' dates, instead of exact dates. Similarly, to avoid recomputing the structural part of each symbolic time process, we will work with *unfoldings*, i.e., structures that contain all symbolic time processes of an STPN but factorize common prefixes. Symbolic unfoldings were introduced for TPNs by SEMENOV and Yakovlev [63] and later used by CHATAIN and Jard [15]. In this section, we show how to unfold STPNs with a blocking semantics, and how to extract symbolic time processes from this unfolding. Our aim is to find the minimal structure that encompasses prefixes of all symbolic time processes that embed a schedule of known duration. We show (theorem 6.1 in this section) that if a system cannot execute arbitrary large sets of events without necessarily progressing time, then unfolding up to some bounded depth is sufficient.

Definition 6.6 (time progress)
An STPN $\mathcal{N}$ guarantees *time progress* iff the following condition is met:

$$\exists \delta \in \mathbb{Q}_{>0}, \forall t \in T, \forall u = \langle t_1, d_1 \rangle \cdots \langle t^i, \theta_0 \rangle \cdots \langle t^{i+1}, \theta_1 \rangle \cdots \in \mathscr{L}(\mathcal{N}) \text{ with } i \in \mathbb{N} : \delta \leq \theta_1 - \theta_0,$$

i.e., there exists a minimum strictly positive duration δ that must elapse before two successive firings of the same transition. ◊

Time progress is close to nonzenoness property, and is easily met (e.g., if no transition has an earliest firing time of 0, or if there is no sequence of transitions that can be repreated and take 0 time units,...). The example of Figure 6.4 does not guarantee time progress as, according to STPN semantics, this net allows an arbitrary number of firings of transition t_0 to occur at date 0. However, such specification can be considered as ill-formed. In train networks, some time must elapse between two consecutive departures of a train from a station, as well as between two consecutive departures or arrivals of the same train. Hence, URSs models ensure properties that are stronger than time progress property (transition can only fire a minimum amount of time after being enabled). More generally, many STPNs modeling real-life cases exclusively contain transitions with rational intervals of the form $[\alpha, \beta]$ or $[\alpha, +\infty)$, where $\alpha > 0$, which is sufficient to guarantee time progress. An interesting consequence of time progress is that any execution of duration Δ of an STPN that guarantees time progress is a sequence of at most $|T| \cdot \lceil \frac{\Delta}{\delta} \rceil$ transition firings. It means that for systems such as URSs that run for a predetermined period (e.g., a metro system usually operates from 05:00 to 01:00), it is sufficient to consider behaviors of a model up to a **bounded horizon** to address their properties over that period and, in particular, check realizability.

As in processes, unfoldings will contain occurrences of place fillings (a set of conditions B), and occurrences of transition firings (a set of events E). We use the same recursive notations for conditions and events: $b \triangleq \langle {}^\bullet b, \mathsf{pl}(b) \rangle$ and $e \triangleq \langle {}^\bullet e, \mathsf{tr}(e) \rangle$, and a dummy initial event $\perp$ to define the set B_0 of initial conditions in an unfolding. We also use relations $\prec$ and $\preceq$ as defined for processes. The main difference between processes and unfoldings is that conditions have a single successor event in time process, and several successor events in unfoldings (branching).

The rest of the chapter addresses realizability in the following way:

- We first define a notion of structural unfolding and show how to extract the untimed structure of time processes from it.

- We then decorate these processes to obtain symbolic processes, i.e., processes with constraints on their associated variables. Solutions to these constraints define execution dates of events.
- we show that all timed processes executable in a bounded duration Δ can be captured by symbolic processes of an unfolding developed up to a bounded depth
- We use this property to verify realizability, i.e., verify that some embedding from a schedule S to the finite set of symbolic processes exists
- We then give tools to compute the probability of time process that embed S.

6.3.1 Structural unfolding

To define structural unfoldings, we first recall the notions of occurrence nets, branching processes, and recall the unfolding algorithm used by ESPARZA et al. [26].

Definition 6.1. An *occurrence net* is an acyclic Petri net $ON = (B, E, F, Cut_0)$ where the elements of B are conditions and those of E are events. and $Cut_0 \subseteq B$ such that: As ON is acyclic, and hence $<_F \triangleq F^+$ and $\leq_F \triangleq F^*$ are strict and weak partial orders; We say that two events e, e' are in *conflict* and write $e \sharp e'$ iff there exists $f \leq_F e, f' \leq_F e'$ such that $f \neq f'$, and ${}^\bullet f \cap {}^\bullet f' \neq \emptyset$. We furthermore impose that:

- $\forall e \in E : \neg(e \# e)$ (no event is in conflict with itself);
- $\forall b \in B, |{}^\bullet(b)| \leq 1$ (every condition has a unique predecessor);
- ON is finitary: for all $x \in E \cup B$, the set $Past(x) \triangleq \{y \mid y \leq x\}$ is finite; and
- Cut_0 contains exactly the $<$-minimal nodes of ON.

Definition 6.7 (Branching processes)
A *branching process* [25] of a STPN $\mathcal{N} = (P, T, A, \lambda)$ is a triple $U = (ON, h, \lambda')$ where $ON = (B, E)$ is an occurrence net, h is a homomorphism from B to P and from E to T and $\forall e \in E, \lambda'(e) = \lambda(h(e))$. A *process* of a net $\mathcal{N}$ is a branching process of $\mathcal{N}$ such that for every condition $b \in B$, $|(b)^\bullet| \leq 1$, or equivalently, such that E is a configuration (it is a causal net). ◊

If $BR_1 = (B_1, E_1, \hat{F}_1, Cut_0, h_1, \lambda'_1)$ and $BR_2 = (B_2, E_2, \hat{F}_2, Cut_0, h_2, \lambda'_2)$ are two branching processes of $\mathcal{N}$, BR_1 is a *prefix* of BR_2 iff $E_1 \subseteq E_2$, and $\hat{F}_1, h_1, \lambda'_1$ are the respective restrictions of $\hat{F}_2, h_2, \lambda'_2$ to B_1 and E_1. The *unfolding* of $\mathcal{N}$, denoted by $\mathscr{U}(\mathcal{N})$, is the maximal branching process w.r.t. the prefix relation. This unfolding can be infinite, but in this chapter, we will only consider finite structures and unfold nets up to a certain depth.

Definition 6.8 (causal past and closure)
The *causal past* of a condition $b \in B$ is a set $\uparrow b \triangleq \{b' \in B \mid b' \preceq b\}$, i.e., all predecessor conditions of b. A set of conditions $B' \subseteq B$ is *causally closed* iff $\forall b \in B', \uparrow b \subseteq B'$. These definitions extend to events. ◊

Definition 6.9 (preprocess)
A *preprocess* of a finite branching process $\mathscr{U} = \langle B, E \rangle$ is a pair $\langle B', E' \rangle$ such that $B' \triangleq {}^\bullet E' \cup E'^\bullet$, and $E' \subseteq E$ is a *maximal* (i.e., there is no larger preprocess containing B' and E'), *causally closed* and *conflict free* set of events. $\Sigma_{PP}(\mathscr{U})$ denotes the set of preprocesses of $\mathscr{U}$. ◊

Preprocesses of an unfolding of an STPN $\mathcal{N}$ represent potential executions of $\mathcal{N}$ when timing constraints are forgotten. We use the term preprocess to emphasize that these nets only extracts events and conditions, but is not yet a time process, as there might be no valid dating function for it.

Definition 6.10 (cuts in preprocesses)
A *cut* of a preprocess is a set of concurrent conditions. As they originate from a preprocess, these conditions have no conflicting events in their causal past. They represent place contents that can be consumed by the next firable transitions at some point in an execution. Given a preprocess $\langle B,E\rangle$, the set of all its possible cuts is denoted $Cuts(B,E)$. ◊

Unfolding a Petri net simply consists in successively appending events and resulting conditions to cuts of already built processes. Following the work of McMillan [51] and Esparza et al. [26], we inductively build unfoldings $\mathcal{U}_0, \mathcal{U}_1 \ldots$. Each step k adds new events, and their postset, to the preceding unfolding $\mathcal{U}_{k-1}$. We start with the initial unfolding $\mathcal{U}_0 \triangleq \langle B_0, \varnothing\rangle$ where $B_0 \triangleq \{\langle \bot, p\rangle \mid p \in m_0\}$. The induction step is defined as follows:

Algorithm 6.1: An unfolding algorithm

input : an unfolding $\mathcal{U}_k \triangleq \langle B_k, E_k\rangle$ (obtained at step k)
output: an unfolding $\mathcal{U}_{k+1}$
1 Build the set $Cuts(\mathcal{U}_k)$ of cuts of $\mathcal{U}_k$
2 $Cuts(\mathcal{U}_k) = \{C \subseteq B_k \mid \exists \langle X,Y\rangle \in \Sigma_{\text{PP}}(\mathcal{U}_k), C \in Cuts(X,Y)\}$
3 Build the set of possible events
$\hat{E} = \{\langle B,t\rangle \in (2^{B_k} \times T) \setminus E_k \mid \exists C \in Cuts(\mathcal{U}_k), B \subseteq C \wedge {}^\bullet t = \mathsf{pl}(B)\}$
4 Build the conditions produced by possible events $\hat{B} = \{\langle\langle B,t\rangle, p\rangle \in \hat{E} \times t^\bullet\}$.
5 **return** $\mathcal{U}_{k+1} = \langle B_k \cup \hat{B}, E_k \cup \hat{E}\rangle$

Intuitively, the set of possible events $\hat{E}$ contains occurrences of transitions whose preset is contained in a cut of a preprocess of $\mathcal{U}_k$, and $\hat{B}$ adds the conditions produced by $\hat{E}$. If one wants to restrict the construction of the unfolding to obtain a structure of bounded depth, computation of $\hat{E}$ can be easily adapted to contain only events at depth smaller than some given K. In such a case, the construction stops with a finite unfolding $\mathcal{U}_K$.

Definition 6.11 (competition)
We say that two events are *in competition* and write $e \curlyvee e'$ when they fill a common place, i.e., $\mathsf{tr}(e)^\bullet \cap \mathsf{tr}(e')^\bullet \neq \varnothing$ ◊

The structural unfolding of an STPN does not consider timing issues nor blockings. However, existence of competing events may result in unsatisfiable constraints on occurrence dates of events (one cannot, for instance, fire two competing events at any date). Hence, a preprocess of $\Sigma_{\text{PP}}(\mathcal{U}_K)$ need not be the untimed version of a time process obtained from a word in $\mathcal{L}(\mathcal{N})$. Indeed, urgent transitions can forbid firing of other conflicting transitions. Similarly, blockings prevent an event from occurring as long as a condition in its postset is filled. They may even prevent events in a preprocess from being executed if a needed place is never freed. However, time processes of an STPN $\mathcal{N}$ can be built from processes of its untimed unfolding. We will show later that, once constrained, time processes of $\mathcal{N}$ are only prefixes of preprocesses in $\Sigma_{\text{PP}}(\mathcal{U}_K)$ with associated timing function that satisfy requirement on dates that only depend on the considered preprocess. We reintroduce time in unfoldings by attaching constraints to events and conditions of preprocesses.

6.4 Constraints to introduce time in processes

The unfolding of a net only consider causal dependencies and conflicts among transitions occurrences. To introduce time and competition in unfoldings, we introduce variables as follows:

- **Condition variables:** We associate to each condition $b \in B$ positive real valued variables $\mathsf{dob}(b)$ and $\mathsf{dod}(b)$ that respectively represent some i^{th} date of birth of a token in place $\mathsf{pl}(b)$, and the date at which this token is consumed.
- **event variables:** we associate to each event $e \in E$ positive real valued variables $\mathsf{doe}(e)$, $\mathsf{dor}(e)$ and $\theta(e)$ that respectively define the dates of the i^{th} enabling, the date at which firability holds, and the firing date of transition $\mathsf{tr}(e)$ represented by event e.

We denote by $\mathrm{Var}(B,E)$ the set of variables for a branching process with conditions b end events E. $\mathrm{Var}(B,E) \bigcup_{b\in B} \mathsf{dob}(b) \cup \mathsf{dod}(b) \cup \bigcup_{e\in E} \mathsf{doe}(e) \cup \mathsf{dor}(e) \cup \theta(e)$. A *constraint* over $\mathrm{Var}(B,E)$ is a boolean combination of atoms of the form $x \bowtie y$, where $x \in \mathrm{Var}(B,E)$, $\bowtie \in \{<,>,\leq,\geq\}$ and y is either a variable from $\mathrm{Var}(E,B)$ or a real constant value. A set of constraints C over a set of variables V is *satisfiable* iff there exists at least one valuation $v : V \to \mathbb{R}$ such that replacing each occurrence of each variable x (and y) by its valuation $v(x)$ yields a tautology. We denote by $\mathrm{Sol}(C)$ the set of valuations that satisfy C.

Let $\mathscr{U}_K \triangleq \langle B_K, E_K \rangle$ be the structural unfolding of an STPN $\mathscr{N}$ up to depth K, and let $B \triangleq {}^\bullet E \cup E^\bullet$ be a set of conditions contained in B_K, and $E \subseteq E_K$ a conflict free and causally closed set of events. We define $\Phi_{B,E}$ as the set of constraints attached to conditions in B and events in E, (i.e., defined over a set of variables $\mathrm{Var}(B,E)$), assuming that executions of $\mathscr{N}$ start at a fixed date dt_0. Constraints in $\Phi_{B,E}$ are set to guarantee:

- **net constraints:** occurrence dates of events are compatible with the earliest and latest firing times of transitions in $\mathscr{N}$,
- **causal precedence:**if event e precedes event e', then $\theta(e) \leq \theta(e')$,
- **no overlapping conditions:** if b, b' represent occurrences of the same place, then intervals $[\mathsf{dob}(b), \mathsf{dod}(b)]$ and $[\mathsf{dob}(b'), \mathsf{dod}(b')]$ have, at most, one common point,
- **urgency**
  an urgent transition with empty postset fires if no other urgent transition fires before; and, in particular, for every event $e \in E$, there is no event that becomes firable and urgent before e.

Conditions

Let us first define the constraints associated with each condition in $b \in B$. Recalling that variable $\mathsf{dob}(b)$ represents the date at which condition b is created, $\Phi_{B,E}$ must first impose that for every condition $b \in B_0$, $\Phi_{B,E}$ contains a constraint of the form $\mathsf{dob}(b) = dt_0$. For every other condition $b = \langle e, p \rangle \in B \setminus B_0$, as the date of birth of b is exactly the occurrence date of e, we impose that $\Phi_{B,E}$ contains a constrint $\mathsf{dob}(b) = \theta(e)$. Despite this equality, we will use both variables $\theta(e)$ and $\mathsf{dob}(b)$ to enhance readability. Recall that $\mathsf{dod}(b)$ is a variable that designates the date at which a place is emptied by some transition firing, $\mathsf{dod}(b)$ is hence the occurrence date of an event that has b as predecessor. Within a conflict free set of events, this event is unique.

In the considered subset of conditions B, several conditions $b, b', b'' \ldots$ may represent fillings of the same place ($\mathsf{pl}(b) = \mathsf{pl}(b') = \mathsf{pl}(b'') = \ldots$). The set B can, thus, be seen as a

partition $B = \bigcup_{p \in P} B_p$, where $B_p = \{b \in B \mid \mathsf{pl}(b) = p\}$, i.e. each condition in B_p represents a particular filling of place p.

Definition 6.12 (siblings and family)
Given a preprocess $\langle B, E \rangle$ and a condition $b \in B$, the *siblings* of b are the conditions associated with the same place as b and that are concurrent with b. More formally, we denote by $\mathrm{Sib}(B, E, b) \triangleq \{b' \in B \setminus \{b\} \mid \mathsf{pl}(b) = \mathsf{pl}(b') \wedge b \parallel b'\}$ the set of siblings of b in preprocess $\langle B, E \rangle$.

We also define the *family* of a set of conditions $X \in B$ as the union of all their siblings. We denote the family of a set of conditions X in a preprocess $\langle B, E \rangle$ by $\mathrm{Fam}(B, E, X) \triangleq \bigcup_{b \in X} \mathrm{Sib}(b)$. We will simply write $\mathrm{Sib}(b)$ and $\mathrm{Fam}(X)$ when B and E are clear from the context. ◊

Due to elementary semantics, all conditions in a particular subset B_p must have disjoint existence dates, that is, for every $b, b' \in B_p$ with $b \neq b'$, the intersection between $[\mathsf{dob}(b), \mathsf{dod}(b')]$ and $[\mathsf{dob}(b'), \mathsf{dod}(b)]$ is either empty, or limited to a single value. We will write no-ov(b, b') (for *no overlapping*) to define a constraint that imposes such conditions on dates of birth and death of b and b'. More formally,

$$\text{no-ov}(b, b') \triangleq \begin{cases} \mathsf{dod}(b) \leq \mathsf{dob}(b') \vee \mathsf{dod}(b) \leq \mathsf{dob}(b) & \text{if } b^\bullet \neq \varnothing \wedge b'^\bullet \neq \varnothing, \\ \mathsf{dod}(b) \leq \mathsf{dob}(b') & \text{if } b^\bullet \neq \varnothing \wedge b'^\bullet = \varnothing, \\ \mathsf{dod}(b') \leq \mathsf{dob}(b) & \text{otherwise.} \end{cases}$$

To enforce elementary semantics, constraint no-ov(b, b') must hold for every pair of conditions b, b' such that $b \in B_p$. Note that if $b \preceq b'$, then the constraint among events and transitions immediately ensures $\mathsf{dob}(b) \leq \mathsf{dod}(b) \leq \mathsf{dob}(b') \leq \mathsf{dod}(b')$, so one needs not additional constraints to avoid overlapping between existence dates of b and b'. However, for pairs of concurrent conditions, no causal relation enforce automatically that existence dates of conditions do not overlap. Hence, to define processes with consistent timing, we have to ensure the existence dates of every condition does not collide with the existence of all its siblings. Hence $\Phi_{B,E}$ must contain a constraint of the form nob(b) (for *no blocking*) for every condition in B.

$$\text{nob}(b) \triangleq \bigwedge_{b' \in \mathrm{Sib}(b)} \text{no-ov}(b, b').$$

In words, condition b does not hold during the validity dates of any concurrent condition representing the same place. In particular, a time process of $\mathcal{N}$ cannot contain two maximal conditions with the same place.

Events

Let us now consider the constraints attached to events. An event $e \triangleq \langle X, t \rangle$ is an occurrence of a firing of transition t that needs conditions in X to be fulfilled to become enabled. Calling $\mathsf{doe}(e)$ the *date of enabling* of e, we have $\forall e \in E$:

$$\mathsf{doe}(e) = \max_{b \in X} \mathsf{dob}(b).$$

We define the *date of readiness* $\mathsf{dor}(e)$ of event e, the date at which time-to-fire of transition

t elapses totally (i.e., becomes null). As the initial time-to-fire is samples from interval $[\mathsf{eft}(t), \mathsf{lft}(t)]$ an occurrence of a transition becomes firable at least, $\mathsf{eft}(t)$ time units and, at most, $\mathsf{lft}(t)$ time units after the date of enabling $\mathsf{doe}(e)$. Hence, $\Phi_{B,E}$ must contain the following constraint const-ready(e) for every event $e \in E$, where:

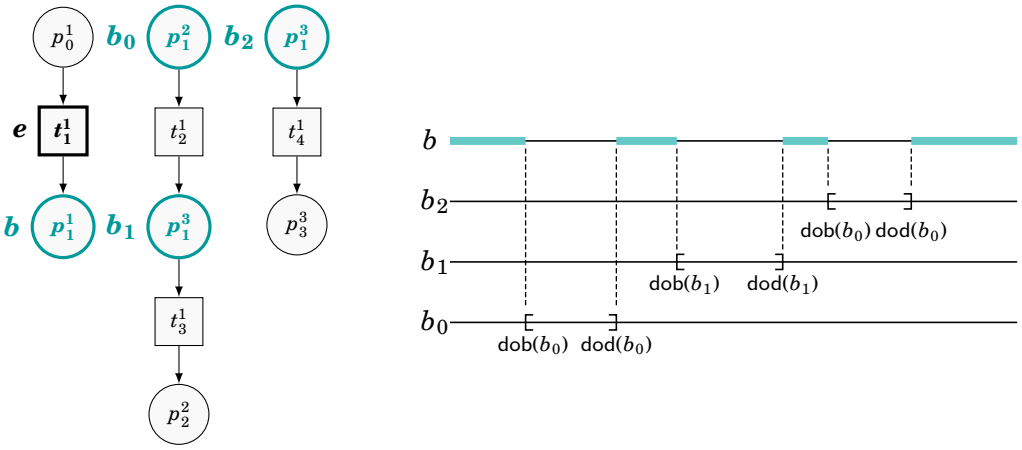
$$\text{const-ready}(e) \triangleq \mathsf{doe}(e) + \mathsf{eft}(t) \leq \mathsf{dor}(e) \leq \mathsf{doe}(e) + \mathsf{lft}(t).$$

As a consequence of elementary semantics, execution of e does not always immediately occur when e is ready. It occurs after $\mathsf{dor}(e)$, and as soon as the places filled by e are empty, i.e., e occurs at a *date of firing* $\theta(e)$ at which no place in $t^\bullet$ is occupied. An accurate date of firing of e w.r.t. the blocking semantics is guaranteed by integrating for each event $e \in E$ the constraint acc-firing(e) to $\Phi_{B,E}$, where:

$$\text{acc-firing}(e) \triangleq \theta(e) = \mathsf{dob}(b) \wedge \mathsf{nob}(b).$$

Last, as STPN semantics is urgent, once ready, e has to fire at the earliest possible date. This is encoded by the constraint

$$\theta(e) = \max\big(\mathsf{dor}(e), \min\big(\mathbb{R}_{\geq 0} \setminus \bigcup_{b \in \mathrm{Fam}(e^\bullet)} (\mathsf{dob}(b), \mathsf{dod}(b))\big)\big).$$



(a) (b)

Figure 6.5: Constraints on dates of birth of tokens in a shared place.

Figure 6.5 shows the effect of blocking and possible firing dates for some event with a condition b in its postset. The top of the figure is a part of a preprocess, with conditions b, b_0, b_1, and b_2 referring to the same place p_1. Suppose that values of variables $\mathsf{dob}(b_i)$ and $\mathsf{dod}(b_i)$ for all $i \in \{0,1,2\}$ are already known. The situation is depicted by the drawing at the bottom of Figure 6.5. Horizontal lines represent real axis, and line portion between brackets represent intervals $[\mathsf{dob}(b_i), \mathsf{dod}(b_i)]$. According to the considered preprocess, we have $\mathrm{Sib}(b) \triangleq \{b_0, b_1, b_2\}$. Then $[\mathsf{dob}(b), \mathsf{dod}(b)]$ have to be fully contained into one of the thick segments of the Figure. An event with b in its postset, as event e in the depicted preprocess, can only occur at dates contained in these thick segments.

Similarly, event e must occur before all its conflicting events. If an event e' in conflict

with e is executed, at least one condition in ${}^\bullet e$ is consumed, and e cannot appear in a time process containing e'. Hence, for every event $e \in E$, $\Phi_{B,E}$ must include the additional constraint $\bigwedge_{e' \sharp e} \mathrm{nmu}(e,e')$ (for *not more urgent*) to guarantee that there exists no other event that is forced to occur before e due to urgency. We define $\mathrm{nmu}(e,e')$ as the following constraint:

$$\mathrm{nmu}(e,e') \triangleq \theta(e) \geq \mathsf{doe}(e') + \mathsf{lft}(\mathsf{tr}(e')) \implies \mathrm{tiled}(e,e') \vee \bigvee_{e'' \| e} \mathrm{preempt}(e',e'')$$

where

- $e'' \| e$ refers to events that are concurrent with e in the considered set of events E;
- $\mathrm{preempt}(e',e'') \triangleq \theta(e'') \leq \min((\mathsf{doe}(e') + \mathsf{lft}(\mathsf{tr}(e')), \theta(e)) \cap \mathsf{fre}(e'))$ is the *preemption* condition, and it means that e'' disabled e' by consuming a condition in ${}^\bullet e''$;
- $\mathsf{fre}(e') \triangleq \mathbb{R}_{\geq 0} \setminus \{[\mathsf{dob}(b), \mathsf{dod}(b)] \mid \exists b' \in e'^\bullet, b \in I(b')\}$ is the set of *free* intervals, i.e., in which places attached to conditions in $e'^\bullet$ are empty; and
- $\mathrm{tiled}(e,e') \triangleq \mathsf{fre}(e') \cap [\mathsf{doe}(e') + \mathsf{lft}(\mathsf{tr}(e')), \theta(e)] = \varnothing$.

Constraint $\mathrm{nmu}(e,e')$ means that if e' is in conflict with e, then at least one condition in ${}^\bullet e'$ is consumed before e' can fire, or if e' becomes firable before e fires, the urgent firing of e' is delayed by blockings so that e can occur. As for constraint attached to blockings, $\mathrm{nmu}(e,e')$ can be expressed as a boolean combination of inequalities. One can also notice that $\mathrm{nmu}(e,e')$ can be expressed without referring to variables attached to event e' nor $e'^\bullet$, as $\mathsf{doe}(e') = \max_{b_i \in {}^\bullet e'} \mathsf{dob}(b_i)$ and the intersection of $\mathrm{Sib}(b)$ and $e'^\bullet$ is empty.

For causally closed sets of events and conditions $B \cup E$ contained in some preprocess of $\mathscr{U}_K$, the constraint $\Phi_{B,E}$ applying on events and conditions of $B \cup E$ is now defined as $\Phi_{B,E} \triangleq \bigwedge_{x \in B \cup E} \Phi_{B,E}(x)$ where:

$$\forall b \in B, \Phi_{B,E}(b) \triangleq \begin{cases} \mathsf{dob}(b) = dt_0 & \text{if } b \in B_0 \text{ and } b \text{ is maximal,} \\ \mathsf{dob}(b) = dt_0 \wedge \mathsf{dob}(b) \leq \mathsf{dod}(b) & \text{if } b \in B_0, \\ \mathsf{dob}(b) = \theta({}^\bullet b) & \text{if } b \notin B_0 \text{ and } b \text{ is maximal,} \\ \mathsf{dob}(b) = \theta({}^\bullet b) \wedge \mathsf{dob}(b) \leq \mathsf{dod}(b) & \text{otherwise.} \end{cases} \\ \wedge \quad \mathrm{nob}(b)$$

$$\forall e \in E, \Phi_{B,E}(e) \triangleq \begin{bmatrix} \mathsf{doe}(e) = \max_{b \in {}^\bullet e} \mathsf{dob}(b) \\ \wedge \quad \mathsf{doe}(e) + \mathsf{eft}(\mathsf{tr}(e)) \leq \mathsf{dor}(e) \leq \mathsf{doe}(e) + \mathsf{lft}(\mathsf{tr}(e)) \\ \wedge \quad \mathsf{dor}(e) \leq \theta(e) \\ \wedge \quad \bigwedge_{b \in {}^\bullet e} \mathsf{dod}(b) = \theta(e) \\ \wedge \quad \bigwedge_{b \in e^\bullet} \theta(e) = \mathsf{dob}(b) \\ \wedge \quad \bigwedge_{e' \sharp e} \mathrm{nmu}(e,e') \end{bmatrix}$$

Let us now address maximality of symbolic prefixes w.r.t. urgent events occurrences.

Definition 6.13 (symbolic prefix)
Given a preprocess $\mathrm{PP} \triangleq \langle B, E \rangle$ of an unfolding $\mathscr{U}_K$, a *symbolic prefix* of PP is a triple $SPP = \langle B', E', \Phi_{B',E'} \rangle$ where $E' \subseteq E$ is a causally closed set of elements, and $B' \triangleq {}^\bullet E' \cup E'^\bullet$. $\Phi_{B',E'}$ is the set of all constraints that help fulfilling net constraints, causality, overlapping, and urgency. ◊

An important point to notice is that in a symbolic prefix $\langle B,E,\Phi_{B,E}\rangle$, $\langle B,E\rangle$ is a preprocess, i.e. E and B are causally closed sets of events and conditions. However, the constraint $\Phi_{B,E}$ can be unsatisfiable, i.e. there might be no timed non-blocking realization of $\langle B,E,\Phi_{B,E}\rangle$. Hence a symbolic prefix is not always a symbolic characterization for a set of time processes.

Let SPP $\triangleq \langle B',E',\Phi_{B',E'}\rangle$ be a symbolic prefix of preprocess PP $\triangleq \langle B,E\rangle$. Symbolic process SPP is *maximal w.r.t. urgent events occurrences* iff no more event of PP *must necessarily* belong to SPP. This property of SPP holds if every event $f \in B'^{\bullet} \cap E$ that could have become urgent before the last date of all events in E' was prevented from firing, due to blocking. This property of prefixes can be verified as a property $\Phi_{\max}(f)$ that has to be satisfied for every $f \in B'^{\bullet} \cap E$. Letting $C_f \triangleq \mathsf{pl}^{-1}(f^{\bullet}) \cap B'$ denote the set of conditions of B' whose place appears in the postset of f, SPP is maximal iff for every $f \in B'^{\bullet} \cap E$, the following constraint is not satisfiable:

$$\Phi_{\max}(f) \triangleq \left[\begin{array}{ll} & \Phi_{B',E'} \\ \wedge & \mathsf{eft}(f) + \max_{b\in {}^{\bullet}f} \mathsf{dob}(b) \leq \theta(f) \qquad \text{(a)} \\ \wedge & \bigvee_{X\in 2^{C_f}} \max_{x\in X} \mathsf{dod}(x) \leq \theta(f) \leq \min_{x\in C_f\setminus X} \mathsf{dob}(x) \qquad \text{(b)} \\ \wedge & \theta(f) \leq \max_{e'\in E'} \theta(e') \qquad \text{(c)} \end{array}\right.$$

Intuitively, $\Phi_{\max}(f)$ means that event f that is not in the symbolic process: (a) becomes urgent, (b) is not blocked by conditions in B', and (c) has to fire before the execution of the last event in E'. If $\Phi_{\max}(f)$ is satisfiable, then f should appear in the process.

We can now define symbolic processes, and show how instantiation of their variables define time processes of $\mathcal{N}$. Roughly speaking, a symbolic process is a prefix of a preprocess of $\mathcal{U}_K$ (and, hence, a causal net) decorated with a *satisfiable* set of constraints on occurrence dates of events. Before formalizing symbolic processes, let us highlight three important remarks.

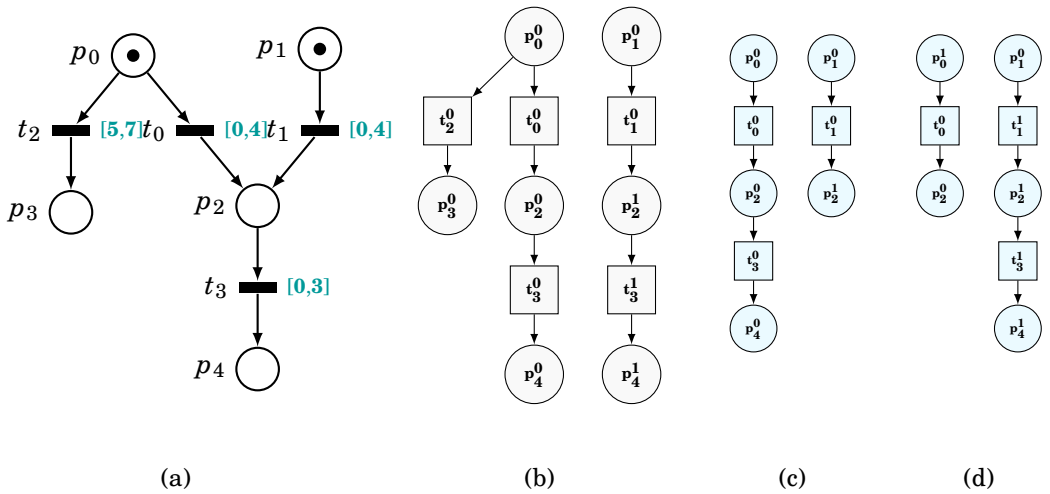
Remark 6.1 (constraints). An unfolding up to depth K misses some constraints on occurrence dates of events due to blockings by conditions that do not belong to $\mathcal{U}_K$ but would appear in some larger unfolding $\mathcal{U}_{K'}$, with $K' > K$, We will however show (cf. Proposition 6.1 and Theorem 6.1) that with time progress assumption, unfolding $\mathcal{N}$ up to a sufficient depth guarantees that all constraints regarding events with $\theta(e) \leq D$ are considered. This allows to define symbolic processes representing the time processes of $\mathcal{N}$ that are executable in less than D time units.

Remark 6.2 (event depth). Unfoldings consider depth of events, and not their dates. Hence, if a process contains an event e occurring at some date greater than d, and another event e' that belongs to the same preprocess and becomes urgent before date d, then e' must belong to the process, even if it lays at a greater depth than e.

Definition 6.14 (symbolic process)
A *symbolic process* of an unfolding $\mathcal{U}_K = \langle E_K,B_K\rangle$ is a triple $Sp \triangleq \langle B',E',\Phi_{B',E'}\rangle$ that is is a symbolic prefix of some preprocess PP $\triangleq \langle B,E\rangle$ of $\mathcal{U}_K$ in which $\Phi_{B',E'}$ is satisfiable, and such that E' is maximal w.r.t. urgent events occurrence in PP. ◊

Intuitively, a symbolic process is a prefix where all transitions that had to be fire have

been integrated the the set of events. As already mentioned, it is not sufficient to equip a preprocess $\langle B,E\rangle$ of $\mathscr{U}_K$ constraint $\Phi_{B,E}$ to obtain a symbolic process. Indeed, some events in a preprocess might be competing for the same resource, and make $\Phi_{B,E}$ unsatisfiable. Consider for instance the STPN of Figure 6.6-a). Its unfolding is represented in b), and two of its symbolic processes in c) and d). For readability, we have omitted constraints. One can however notice that there exists no symbolic process containing two occurrences of transition t_3, because conditions p_4^0 and p_4^1 are maximal and represent the same place p_4.



(a) (b) (c) (d)

Figure 6.6: An STPN with conflicts and blockings (a), its symbolic unfolding (b), and two of its symbolic processes (c) and (d).

A crux in the construction of symbolic processes of $\mathscr{U}_K$ is to find appropriate maximal and causally closed sets of events with satisfiable constraints. This can be costly: as illustrated by the example of Figure 6.6, satisfiability of constraints is not monotonous: the constraints for processes in Fig 6.6$-c$) and d) are satisfiable. However, adding one occurrence of transition t_3 yields unsatisfiable constraints. Satisfiability of a prefix of size n hence does not imply satisfiability of a larger prefix of size $n+1$. The converse implication is also false: if a constraint associated with a prefix of size n is not satisfiable, appending a new event may introduce blockings that delay urgent transitions, yielding satisfiability of a constraint on a prefix of size $n+1$. So, unsatisfiability of constraints cannot be used as a criterion to stop incremental construction of unfoldings.

Definition 6.15 (execution of a symbolic process)
Let $\mathrm{Sp} \triangleq \langle B,E,\Phi_{B,E}\rangle$ be a symbolic process of an unfolding $\mathscr{U}_K$. An *execution* of Sp is a time process $\mathrm{TP} \triangleq \langle B,E,\theta\rangle$ where θ is a solution of $\Phi_{B,E}$. For a given θ, we denote by $\mathrm{Sp}_\theta \triangleq \langle B,E,\theta\rangle$ the time process obtained from Sp. A tuple $\langle B,E,\theta\rangle$ is a time process of a given unfolding $\mathscr{U}_K$ iff it is an execution of one of its symbolic processes. ◊

Informally, symbolic preprocesses select maximal conflict free sets of events in an unfolding. Symbolic processes extract executable prefixes from symbolic preprocesses, and executions attach dates to events of symbolic processes to obtain time processes.

We shall respectively denote by $\mathscr{SPR}_{\mathrm{Sp}}(\mathscr{U}_K)$ and by $\mathscr{TP}_{\mathrm{TP}}(\mathscr{U}_K)$ the set of symbolic processes and time processes of $\mathscr{U}_K$.

We can now show that under time progress assumption, unfoldings and their symbolic

processes capture the elementary semantics of STPNs, i.e. represent all their time processes. Given an STPN that guarantees time progress with a minimal elapsing of δ time units between successive occurrences of each transition $t \in T$, and given a maximal date D, we want to build an unfolding $\mathscr{U}^D$ of $\mathscr{N}$ that contains all events that might be executed before D, but also all conditions and events which may impact firing dates of these events. We can show that $\mathscr{U}^D$ is finite and that its processes are of depth at most $H = \lceil \frac{D-dt_0}{\delta} \rceil \cdot |T|$.

Let $b \triangleq \langle e, p \rangle$ be a condition of an unfolding $\mathscr{U}_n$ obtained at step n. Let $\mathrm{Sib}^*(b)$ be the set of conditions that may occur in the same process as b, represent the same place, and are not predecessors or successors of b in any unfolding $\mathscr{U}_{n+k}$ obtained from $\mathscr{U}_n$. Clearly, dates of birth and death of conditions in $\mathrm{Sib}^*(b)$ may influence the date of birth and death of b, or even prevent b from appearing in the same process as some conditions in $\mathrm{Sib}^*(b)$. However, in general, $\mathrm{Sib}^*(b)$ need not be finite, and at step n, $\mathrm{Sib}^*(b)$ is not fully contained in a preprocess of $\mathscr{U}_n$. Fortunately, under time progress assumption, we can show that elements of $\mathrm{Sib}^*(b)$ that can influence $\mathsf{dob}(b)$ appear in some bounded unfolding $\mathscr{U}_K$.

Proposition 6.1. *Let $\mathscr{N}$ be an STPN guaranteeing time progress of δ time units (between consecutive occurrences of each transition). For every date $D \in \mathbb{R}_{\geq 0}$ and condition b in an unfolding $\mathscr{U}_n$, there exists $K \geq n$ such that $\{b' \in \mathrm{Sib}^*(b) \mid \mathsf{dob}(b') \leq D\}$ is contained in $\mathscr{U}_K$.*

This proposition means that if some event cannot occur at $\mathsf{dor}(e)$ due to a blocking, i.e., $\theta(e) \neq \mathsf{dor}(e)$, then one can discover all conditions that prevent this firing from occurring in a bounded extension of the current unfolding.

Theorem 6.1. *Let $\mathscr{N}$ be an STPN guaranteeing time progress of δ time units. The set of time processes executable by $\mathscr{N}$ in D time units is a set of prefixes of time processes of $\mathscr{U}_K$, with $K = \lceil \frac{D}{\delta} \rceil \times |T|^2$ containing only events with date less than or equal to D.*

We do not detail the proofs of Proposition 6.1 and Theorem 6.1 here, and refer interested reader to Appendix A.1 and Appendix A.2 for detailed proofs. This theorem is central for realizability: it means that is is sufficient to unfold a net up to a depth that guarantees finding all processes that realize a particular schedule. We explain how in the next section.

6.5 Boolean realizability of schedules

We can now address the question of realizability of a high-level description of operations (a schedule S) by a system (described by a STPN $\mathscr{N}$). Considered in a purely boolean setting, this question can be rewritten as: *"Is there an execution of $\mathscr{N}$ that implements S?"* In many cases, a positive answer to this question is not sufficient: as STPNs are equipped with continuous probability distributions, the probability of a particular execution $\mathrm{TP} \triangleq \langle E, B, \theta \rangle$ is always null. A sensible way to address realizability is a quantitative approach requiring that the set of executions of $\mathscr{N}$ implementing S has a strictly positive probability. In this section, we first formalize the notion of realization of a schedule by an execution, and define boolean realizability. We then define probabilistic realizability, and show how an underapproximation of the probability to realize a schedule can be computed using a transient tree construction [39].

First of all, the connection between high-level description of operations in S and their implementation in $\mathscr{N}$ is defined via a realization function.

Definition 6.16 (realization function)
A *realization function* for a schedule S and an STPN $\mathcal{N}$ is a map $r : \mathscr{A} \to 2^T$ that assigns a subset of transitions from T to each letter of $\mathscr{A}$, with $\forall a \in \mathscr{A}, \forall a' \in \mathscr{A} \setminus \{a\} : r(a) \cap r(a') = \varnothing$. ◊

A realization function describes which low-level actions implement a high-level operation of a schedule. Each letter a from $\mathscr{A}$ can be interpreted as an operation performed through the firing of any transition from the subset of transitions $r(a)$. Allowing $r(a)$ to be a subset of T provides some flexibility in the definition of schedules: in a production cell, for example, a manufacturing step a for an item can be implemented by different processes on different machines. Similarly, in a train network, a departure of a train from a particular station in the schedule can correspond to several departures using different tracks, or to departure with different speed profiles, which is encoded with several transitions in an STPN. Realization functions, hence, relate actions in schedules to several transitions in an STPN. The condition $r(a) \cap r(a') = \varnothing$ prevents ambiguity by enforcing each transition to appear, at most, once in the image of r. Note that $r(\mathscr{A}) \subseteq T$, i.e., the realization of a schedule may need many intermediate steps that are depicted in the low-level description of a system, but are not considered in the high-level view provided by a schedule. This allows, in particular, to define schedules that constrain dates for a subset of events, and leave dates of other events free from any constraint. In the context of RTSs, this allows for the verification of realizability of schedules that focus on a subset of stations, e.g., requiring a departure from a chosen station every x minutes during a normal day of operation. Transitions in $r(\mathscr{A})$ are called *realizations* of $\mathscr{A}$.

Definition 6.17 (embedding)
Let $S \triangleq \langle N, \to, \lambda, C\rangle$ be a schedule, $S\mathrm{p} \triangleq \langle B, E, \Phi\rangle$ be a symbolic process of $\mathcal{N}$ and $r : \mathscr{A} \to 2^T$ be a realization function. We say that S *embeds* into $S\mathrm{p}$ (w.r.t. r and d), and write $S \hookrightarrow S\mathrm{p}$ iff there exists an injective function $\psi : N \to E$ such that:

$$\left[\begin{array}{ll} & \forall n \in N : \mathsf{tr}\big(\psi(n)\big) \in r\big(\lambda(n)\big) \ \texttt{(embedding is consistent with labeling)} \\ \wedge & \forall \langle n, n'\rangle \in \to : \psi(n) \preceq \psi(n') \ \texttt{(causal precedence is respected)} \\ \wedge & \nexists f \leq \psi(\min(n)) : \mathsf{tr}(f) \in r(\mathscr{A}) \ \texttt{(embedding starts on 1st compatible events)} \\ \wedge & \forall e \leq f \leq g : e = \psi(n) \wedge g = \psi(n'') \wedge \mathsf{tr}(f) \in r(\mathscr{A}) \ \texttt{(embedding misses...} \\ & \qquad\qquad \Rightarrow \exists n' : f = \psi(n') \wedge n \to^* n' \to^* n'' \ \texttt{...no compatible event)} \end{array}\right.$$

◊

S embeds in $S\mathrm{p}$ iff there is a way to label every node n of S by a letter from $r\big(\lambda(n)\big)$, and obtain a structure that is contained in some restriction of a prefix of $S\mathrm{p}$ to events that are realizations of actions from $\mathscr{A}$, and to a subset of its causal ordering. This way, a process respects the ordering described in S, does not forget actions, and does not insert realizations that are not the image by ψ of any high-level operation between two mapped realizations, or before the image by ψ of minimal nodes in the schedule. Note that there can be several ways to embed S into a process of $\mathcal{N}$.

Definition 6.18 (realizability)
Let d be a dating function for a schedule $S \triangleq \langle N, \to, \lambda, C\rangle$, and r be a realization function. The pair $\langle S, d\rangle$ is *realizable* by $S\mathrm{p} \triangleq \langle E, B, \Phi\rangle$ (w.r.t. r) iff there exists an embedding ψ from S to $S\mathrm{p}$, and furthermore, $\Phi_{\psi,S,d} \triangleq \Phi \wedge \bigwedge_{n\in N} \theta\big(\psi(n)\big) = d(n)$ is satisfiable. $\langle S, d\rangle$ is *realizable* by $\mathcal{N}$ (w.r.t. r and d) iff there exists a symbolic process $S\mathrm{p}$ that realizes S. ◊

Realizability of a schedule $S \triangleq \langle N, \rightarrow, \lambda, C\rangle$ with constraints C stands for realizability of any of dating function d meeting constraints C. Very often, we also address realizability of a schedule with respect to a fixed dating function d. Letting C_ψ denote the conjunction of inequalities obtained by replacing every assignment $\langle i, j\rangle \rightarrow v$ in map C by inequalities $v \leq \theta\big(\psi\left(n_j\right)\big) - \theta\big(\psi(n_i)\big)$, we will say that S is *realizable* by $S\mathrm{p}$ (w.r.t. r) iff there exists an embedding ψ from S to $S\mathrm{p}$, and $\Phi_{\psi,S,C} \triangleq \Phi \wedge C_\psi$ is satisfiable. Similarly, when nodes of schedules are assigned a precise date by a map d, we can define $C_d \triangleq C_\psi \wedge \bigwedge_{n\in N} \theta(\psi(n)) = d(n)$, and $\Phi_{\psi,S,d} \triangleq \Phi \wedge C_d$. We write $S\mathrm{p} \models S$ when S is realizable by $S\mathrm{p}$, and $\mathcal{N} \models S$ when S is realizable by some symbolic process of $\mathcal{N}$.

Finding the set of embedding functions $\Psi \triangleq \{\psi_0, \psi_1, \ldots, \psi_{k-1}\}$ that satisfy the conditions of definition 6.17 is achieved building iteratively injective functions meeting the embedding requirements, by matching at every step minimal yet unexplored nodes of S with minimal unmatched nodes of $S\mathrm{p}$. This construction is depicted in Algorithm 6.2. We will define an embedding function f as a set pairs of the form $\langle n, e\rangle$, interpreted as $f(n) = e$. We define in particular the empty embedding $f_\perp$, that is undefined for every node of N, and will be used as starting point of the algorithm. For a given function f, will write $f \cup \langle n, e\rangle$ to denote the extension of f to $\{n\}$, that associates e to node n, and $f(n')$ to every other node $n' \in \mathrm{dom}(f)$.

Algorithm 6.2: Computation of embeddings of a schedule in a symbolic process

input: a schedule $S \triangleq \langle N, \rightarrow, \lambda, C\rangle$, a symbolic process $S\mathrm{p} \triangleq \langle E, B, \Phi\rangle$;
$\Psi := \varnothing$; `// the set of solutions is initially empty`
$F := \{f_\perp\}$; `// the exploration starts from the undefined map`
while $F \neq \varnothing$ **do**
  choose $f \in F$;
  $\textsc{Min}_{S,f} := \min_{\rightarrow}\big(N \setminus \mathrm{dom}(f)\big)$;
  if $\mathrm{Min}_{S,f} = \varnothing$ **then** ; `// all nodes of` S `have an image in` $\mathscr{E}$
    $\Psi := \Psi \cup \{f\}$; `// ` f `is an embedding`
  else
    $F := F \setminus \{f\}$; `// we will explore extensions of partial embedding` f
    $\textsc{Min}_{\mathscr{E},f} := \min_{\leq}\big(E \setminus \mathrm{image}(f)\big)$;
    $\textsc{Found} := \mathrm{false}$;
    while $\mathrm{Min}_{S,f} \neq \varnothing \wedge \mathrm{Found} = \mathrm{false}$ **do**
      choose $n \in \textsc{Min}_{S,f}$;
      $\textsc{Min}_{S,f} := \textsc{Min}_{S,f} \setminus \{n\}$;
      $\textsc{Cand} := \{e \in \textsc{Min}_{\mathscr{E},f} \mid \mathrm{tr}(e) \in r\big(\lambda(n)\big) \wedge \forall n' \in \mathrm{dpred}_{\rightarrow}(n), f(n') \preceq e\}$;
      if $\mathrm{Cand} \neq \varnothing$ **then**
        $F := F \cup \bigcup_{e\in\textsc{Cand}} \{f \cup \langle n, e\rangle\}$; `// update` f `with pairs` $\langle n, e\rangle$ `that meet the matching criteria and add the new functions to candidate embeddings`
        $\textsc{Found} := \mathrm{true}$;

Algorithm 6.2 computes the set $\Psi_{S,S\mathrm{p}}$ of embeddings of a schedule S in a symbolic process $S\mathrm{p}$. If there exists an embedding $\psi \in \Psi_{S,S\mathrm{p}}$ from S to a symbolic process $S\mathrm{p}$, such that $\Phi_{\psi,S,C}$ (resp. $\Phi_{\psi,S,d}$) is satisfiable, then S is realizable by $S\mathrm{p}$. Realizability, hence,

consists in finding at least one symbolic process of $\mathcal{N}$ with an appropriate embedding in $\psi \in \Psi_{S,\mathrm{Sp}}$. When a maximal occurrence date D for operations in S is provided, and when $\mathcal{N}$ guarantees time progress, such a process is a process of unfolding $\mathcal{U}_K$ (where K is the bound given in Proposition 6.1). We can then compute the set of symbolic processes $\mathcal{SPR}_{\mathrm{Sp}} \triangleq \{\mathrm{Sp}_0, \mathrm{Sp}_1, \ldots, \mathrm{Sp}_{N-1}\}$ of $\mathcal{U}_K$ that embed S, and similarly, for each $\mathrm{Sp}_i \in \mathcal{SPR}_{\mathrm{Sp}}$, the set of possible embedding functions $\Psi_i \triangleq \{\psi_{i,0}, \psi_{i,1}, \ldots, \psi_{i,N_i-1}\}$ for which constraint $\Phi_{\psi,S,C}$ (resp. $\Phi_{\psi,S,d}$) is satisfiable.

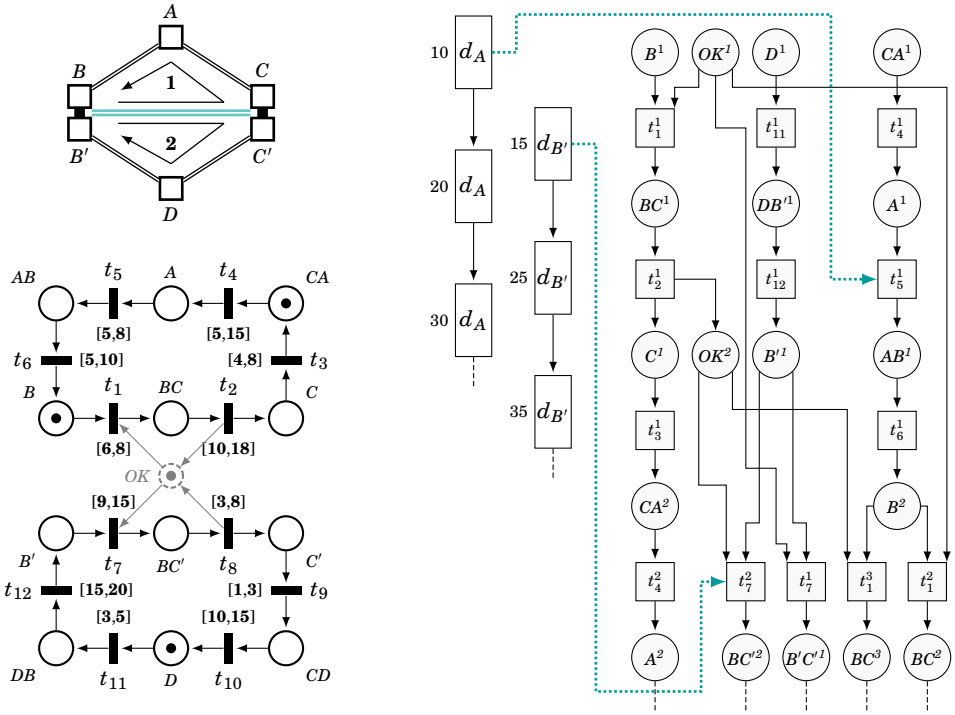


Figure 6.7: Realizability of a schedule for a metro network with two lines and a shared track.

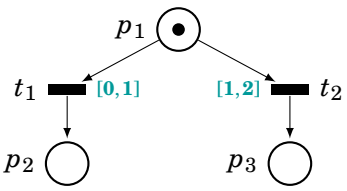
Let us consider the example of figure 6.7. This toy example depicts two train carousels: line 1 serves stations A, B and C, and line 2 serves stations D, B' and C'. Both lines share a common track portion between stations B, C and B', C', and line 1 uses two trains. The upper left picture shows the aspect of both lines and stations, and the bottom left figure an STPN model of this network (we do not show distributions). Stations are represented by places labeled by station names, and track portions between two stations by places labeled by pairs of letters representing the connected stations (e.g., place CA represents the track from C to A). Transitions consuming tokens from a station place represent trains departures, and transitions consuming tokens from a track place are arrivals. A possible required schedule (middle of the figure) is that one train leaves every 10 time units from station A on line 1, starting from date 10, and one train leaves station B' every 10 time units, but starting from date 15. Arrivals of trains and departures from other stations are not represented and are, hence, not constrained. Departures from A are nodes labeled by d_A and departures from B' are nodes labeled by $d_{B'}$. The rightmost part of the figure is a structural unfolding of the net. We set $r(d_A) \triangleq \{t_5\}$ and $r(d_{B'}) \triangleq \{t_7\}$. Note that the topmost occurrence of place OK, that plays the role of a boolean flag in a critical section, can be both consumed by occurrences t_1^1 and t_1^2 of transition t_1, which is a standard conflict. Note also that events t_4^1 and t_4^2 output a token in place A. Even if these

events are not in conflict, due to standard semantics (i.e., nonblocking), their firing dates may influence one another. The way operations of the schedule inject in a process of the net is symbolized by dotted lines. Notice that if t_5^1 has to fire at date 10, then according to intervals attached to transitions, t_4^1 has to fire at date 5. This means that there exists a unique way to guarantee a departure from station A at date 10, which is to sample the smallest running time from C to A and the smallest possible dwell time at station A. The schedule of Figure 6.7 is, thus, realizable. However, the probability of occurrence of this schedule with precise dates is null (again, due to continuity of random variables).

6.6 Probabilistic realizability

The example of Figure 6.7 shows that boolean realizability characterizes an embedding that is consistent with time constraints, but not the probability to realize a schedule. Consider the STPN of Figure 6.8. This simple STPN has two symbolic processes: $S\mathrm{p}_1$ in which transition t_1 fires, and $S\mathrm{p}_2$ in which t_2 fires. The probability of process $S\mathrm{p}_1$ is the probability that a value v_1 sampled to assign a time-to-fire to t_1 is smaller or equal to another value v_2 sampled independently to assign a time-to-fire to t_2. Clearly, the probability that $v_1 \leq v_2$ is equal to the probability that $v_1 \in [0,1]$ (which is equal to 1). The probability of the second process $S\mathrm{p}_2$ is equal to the probability that $v_2 \leq v_1$, but the set of values allowing this inequality is restricted to a single point $\langle v_1, v_2 \rangle = \langle 1,1 \rangle$. Conforming to continuous probability distributions semantics, the probability of this point, and consequently, the probability of executing a time process that is consistent with constraints in $S\mathrm{p}_2$ is null. A schedule S composed of a single node n with a realization function such that $r\big(\lambda(n)\big) \triangleq \{t_2\}$, and a date $d(n) \triangleq 1$ are realizable according to Definition 6.17, but with null probability. Let us slightly change the example of Figure 6.8-a). We now assign interval $[0,3]$ to transition t_1 in the STPN and interval $[1,4]$ to transition t_2. We keep the same schedule S and realization function r, but require that $d(n) = 2$. The probability that t_2 fires from the initial marking is equal to the probability that $v_2 \leq v_1$, which is not null (we explain below and in A.4 how to compute the probability of such domain and the joint probability of v_1 and v_2), and is equal to the joint probability of values of v_1 and v_2 laying in domain $v2 \leq v_1$ depicted by the filled zone in Figure 6.8-b). However, within this continuous domain of possible values, the probability to fire t_2 exactly at date 2 as required by dating function d is still null. Nevertheless, if t_2 is allowed to fire at date 2 with some imprecision α, then the probability to realize the expected schedule is equal to the integration of the joint probability distribution over the domain where $(v_2 \leq v_1) \wedge (2-\alpha \leq v_2 \leq 2+\alpha)$ (represented as a dashed part in Figure 6.8-b), which can be strictly positive if distributions attached to t_1 and t_2 are properly set. Note that requiring dates to be implemented up to some imprecision does not necessarily increase the probability to realize a schedule: in the example of Figure 6.8-*a)* with intervals $[0,1]$ and $[1,2]$, the only way to realize the schedule S mentioned before with date $d(n) = 1$, up to some imprecision, is to execute the unique time process in which t_2 fires at date 1, and the probability of this process is null. More generally, the probability to realize a schedule, when the embedding relation leaves a single possible occurrence date for at least, one event in the chosen symbolic process, is always null.

Boolean realizability is a first step to check that a schedule and an implementation are not totally orthogonal visions of a system. However, examples 6.7 and 6.8 demonstrate that it is not precise enough. They also show that boolean realizability up to imprecision still



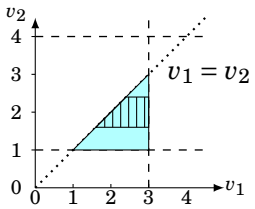


Figure 6.8: *a*) An example STPN *b*) A domain for $\tau(t_1), \tau(t_2)$ allowing firing of t_2, assuming $I(t_1) = [0,3]$ and $I(t_2) = [1,4]$.

allows to consider sets of processes with null probabilities as realizations of a schedule. An accurate notion of realizability should require that schedules embed into symbolic processes of $\mathscr{U}_K$ with strictly positive probability *and* up to some admissible imprecision on dates of events, bounded by some value $\alpha \in \mathbb{Q}_{\geq 0}$. When a schedule is constrained by a precise map d, every operation x in a schedule should now be implemented by an occurrence t_i^j of a transition t at date $\theta(t_i^j) \in [\max(d(x) - \alpha, 0), d(x) + \alpha]$. Once an injection ψ from a schedule $S \triangleq \langle N, \rightarrow, \lambda, C \rangle$ to a symbolic process $S\mathrm{p}$ is found, the constraint to obtain realizability of a dating function d up to imprecision α becomes

$$\Phi_{\psi,S,d\pm\alpha} \triangleq \Phi \wedge \bigwedge_{n \in N} \max(d(n) - \alpha, 0) \leq \theta(\psi(n)) \leq d(n) + \alpha.$$

One can, similarly, require that constraints C in S be realized up to imprecision α, i.e., require that constraints of the form $v \leq d(n_j) - d(n_i)$, imposed by map C, are implemented in the low-level STPN by a time process satisfying a relaxed constraint of the form $v - \alpha \leq \theta\big(\psi(n_j)\big) - \theta\big(\psi(n_i)\big)$. From a practical point of view, this notion of realization up to bounded imprecision is more natural than boolean realizability. Indeed, for systems such as train networks, one cannot expect a schedule to be realized with perfect precision, but rather that differences between realized and scheduled dates be of slight or negligible significance. To measure the probability of realizing a schedule, we define as $\mathbb{P}[S\mathrm{p}]$ the probability of the set of time processes of $S\mathrm{p}$. When an embedding ψ from S to $S\mathrm{p}$ exists, we define as $\mathbb{P}\left[S\mathrm{p} \wedge \mathrm{Sol}(\Phi_{\psi,S,d\pm\alpha})\right]$ the probability of time processes that are realizations of $S\mathrm{p}$ in which dates of events are solutions of $\Phi_{\psi,S,d\pm\alpha}$.

Definition 6.19 (probabilistic realizability)
Let d be a dating function with maximal date D for a schedule $S \triangleq \langle N, \rightarrow, \lambda, C \rangle$, and r a realization function. The pair $\langle S, d \rangle$ is *realizable with nonnull probability* (w.r.t. r) up to imprecision α iff there exists an embedding ψ of S into a symbolic process $S\mathrm{p}$ of $\mathscr{U}_K$ such that $\mathbb{P}(S\mathrm{p} \wedge \mathrm{Sol}(\Phi_{\psi,S,d\pm\alpha})) > 0$. ◊

This definition requires that a symbolic process of $\mathscr{N}$ embeds S, and that the probability that this process is executed and satisfies all timing constraints imposed by the STPN, and by the dating function, is strictly positive. We will now show how to compute this probability using a transient execution tree, as proposed by HORVÁTH et al. [39]. A transient execution tree is a tree whose vertices are *stochastic state classes*.

Definition 6.2 (transient stochastic state class)**.** A *transient stochastic state class* (or class, for short) of an STPN $\mathscr{N}$ is a tuple $\Sigma \triangleq \langle M, C, D, \mathrm{blk}, \mathrm{urg} \rangle$ where M is a marking. For a given class, we define a set of variables X_M with support C representing the possible times-to-fire of transitions enabled by marking M such that for every x in X_M, the elimination of all other variables from C yields a nonempty set of possible values that is

different from {0}; D is a PDF over C, blk is a set of *blocked* transitions, and urg is a set of *urgent* transitions.

Roughly speaking, classes are abstract representations of markings, time domains for sampled values attached to enabled transitions, and of their distributions over the abstract domain. The notion of *state class* was already used for TPN analysis [9, 45], stochastic state classes, hence, only add a probabilistic dimension. In a stochastic state class, transitions that are enabled but not blocked nor urgent are assigned a time-to-fire. We will denote by x_i the variable describing the time to fire associated to a transition t_i, and by X_M the set of all variables attached to enabled transitions in M. The domain C can be described by a set of inequalities of the form $x_i - x_j \bowtie v$ or $x_i \bowtie v$ with $\bowtie \in \{<, >, \leq, \geq, =\}$ and $v \in \mathbb{Q}$. It represents possible values for times-to-fire attached to transitions. Similarly, the distribution D is a PDF defined on $\mathbb{R}_{\geq 0}^{|X_M|}$, giving probability of values of $x_{i_1}, \ldots, x_{i_{|X_M|}}$. When needed, we can also include in X_M, a particular variable x_{age} representing the time elapsed since the beginning of an execution. The probability to fire a particular transition from a class and move to a successor class is computed as an integration over the time domain allowing this transition to fire first. On the example of Figure 6.8 (with intervals $[0,3]$ and $[1,4]$), this corresponds to integration of a joint distribution for values v_1 and v_2 over the filled area.

One can assume that the system under study starts from a known initial marking M_0 with initial known times-to-fire, represented as a point of $\mathbb{R}_{\geq 0}^{|X_{M_0}|}$. The construction of the transient tree starts from class $\langle M_0, C_0, D_0, \text{blk}_0, \text{urg}_0 \rangle$, where C_0 is a single point defining known times-to-fire, D_0 associates probability 1 to the single point in C_0, and 0 to $\mathbb{R}_{\geq 0}^{|X_{M_0}|} \setminus C_0$ (i.e., a Dirac distribution).

Then, the transient tree is built by iterating a construction of successors for already found classes. Let $\langle M, C, D, \text{blk}, \text{urg} \rangle$ be a class such that $\text{urg} \triangleq \varnothing$, and let t_i be an enabled transition in this class. Then, one can compute the domain defined by constraint $C' \triangleq C \wedge \{x_i \leq x_j \mid x_i \neq x_j \in X_M\}$, that imposes that t_i is the first transition to fire, i.e., it has the smallest time-to-fire in the class. If C' is satisfiable, then t_i is effectively firable, and one can compute the probability p_i that t_i fires first, and the successor class $\langle M_i, C_i, D_i, \text{blk}_i, \text{urg}_i \rangle$ reached after firing t_i. First of all, we have $p_i \triangleq \int_{C'} D(x_{i_1} \ldots x_{i_{|X_M|}})$, i.e., the probability of all values for X_M in which x_i is the smallest variable. The successor class $\langle M_i, C_i, D_i, \text{blk}_i, \text{urg}_i \rangle$ is then obtained as follows:

- M_i is the marking $M \setminus {}^\bullet t_i \cup t_i^\bullet$. Sets blk_i and urg_i can be updated as follows: a blocked transition remains blocked if it is enabled by M_i and its postset is not freed by firing t_i. It can also become urgent if its postset is freed, or be disabled if firing t_i removes a token from its preset. Similarly, urgent transitions can become blocked, remain urgent, or be disabled.
- We reuse the technique of LIME and ROUX [45] to compute C_i. We start from C'. We first make a variable change of the form $x_j \triangleq x_i + x'_j$ (to consider the fact that all times-to-fire are decreased by the value of x_i). We, then, eliminate all variables associated to transitions disabled by firing t_i (for instance using the FME method). After that, we add new constraints of the form $x'_k \in I(t_k)$, for every newly enabled transition t_k. This represents the fact that a new time-to-fire is sampled. Last, we rename all x'_k variables to x_k to obtain C_i.
- D_i is obtained in an almost similar way, using the procedure given by HORVÁTH et al. [39]. We have already computed the probability p_i to fire t_i from $\langle M, C, D, \text{blk}, \text{urg} \rangle$.

As t_i fires before any other transition from class $\langle M, C, D, \text{blk}, \text{urg}\rangle$ we first compute a new distribution D^a defined over C' such that $D^a(X_M) \triangleq \frac{D(X_M)}{p}$ that conditions values of variables in $X_M^a \triangleq X_M \setminus \{x_i\}$ knowing that x_i has the smallest value in X_M. The next step is to build the distribution of probabilities once x_i is eliminated, i.e., a distribution $D^b(X_M^a)$ computed as $D^b(X_M^a) \triangleq \int_{\min_i}^{\max_i} D^a(\{x_j + x_i \mid i \neq j\})$, where $\min_i$ is the minimal value of x_i in C' and $\max_i$ its maximal value. This integration is repeated for every variable attached to a transition disabled by firing of t_i. The last step consists in integrating the newly created variables and their distributions to D^b. Let $X_{M_i} \setminus X_M \triangleq \{x_{k_1} \dots x_{k_q}\}$, where $t_{k_1} \dots t_{k_q}$ are newly enabled transitions. As the value of each x_{k_j} represents a newly sampled time-to-fire, it does not depend on former values of variables, we have $D_i \triangleq D^b \cdot F(x_{k_1}) \dots F(x_{k_q})$. D_i is defined over domain C_i defining possible values of variables in X_{M_i}.

The construction of a successor class when $\text{urg} \neq \varnothing$ follows similar lines, i.e., consists in computing successor marking, domain and distribution. The main difference is that an urgent transition necessarily has a time-to-fire equal to 0. Futhermore, it may compete with other urgent transitions. The probability to fire $t_i \in \text{urg}$ is, hence, $\mathbb{P}_{\text{fire}}(t_i) \triangleq \mathcal{W}(t_i) \,/\, \sum_{t_j \in \text{urg}} \mathcal{W}(t_j)$.

One can iteratively compute successors of classes to obtain a transient execution tree. As our STPN is bounded, the number of markings and domains that can be generated inductively at construction time is finite (as proved by BERTHOMIEU and DIAZ [9]). Urgent and blocked transitions are also finite subsets of T. The distributions attached to transitions of STPNs are expolynomial functions. Beyond their expressive power, expolynomial functions are closed under projections, integrations, or multiplication. Furthermore, (joint) distributions of clock values in a node can always be encoded as expolynomial functions. This way, one can iteratively build a tree whose nodes contain markings, classes and expolynomial distributions over these classes. However, as shown by HORVÁTH et al. [39], the number of distributions that can be iteratively computed need not be finite. However, as time progress is guaranteed, one can limit the construction to a bounded horizon. In our case, we can be even more directive, and guide the tree construction to match executions of a particular symbolic process Sp. Indeed, we can consider only executions of a tree that are executions of Sp by remembering at every step which transitions have been executed, and forbidding any transition that is not among the possible next ones.

Details on the construction of this transient tree are provided in Appendix A.3.

After this transient tree construction, the (sum of) probabilities attached to paths of the tree can be used to compute the probability of properties such as safety of a system within a bounded horizon. In our case, the sum of probabilities of all paths that end with the execution of a chosen symbolic process gives the probability to realize this process. If this probability is not null, then there is a positive probability to realize the considered schedule. Note that the computed value is only *a lower bound* for the exact probability to realize a schedule: indeed there can be more than one process realizing a schedule. However, computing the exact probability is rather involved, as distinct realizations of a schedule are not necessarily independent.

Consider a transient tree that collects all symbolic executions of a particular process Sp. Each of its paths

$$\rho \triangleq \langle M_0, C_0, D_0, \text{blk}_0, \text{urg}_0\rangle \xrightarrow{t_1, p_1} \langle M_1, C_1, D_1, \text{BLK}_1, \text{URG}_1\rangle \dots \xrightarrow{t_k, p_k} \langle M_k, C_k, D_k, \text{blk}_k, \text{urg}_k\rangle$$

defines a feasible set of executions of $S\mathrm{p}$ by $\mathcal{N}$. The probability of execution ρ is the product $p_1 \cdot p_2 \cdots p_k$. By summing up probabilities of all paths of the transient tree that are executions of $S\mathrm{p}$, one obtains the probability of $S\mathrm{p}$.

This immediately gives us an informal algorithm to check probabilistic realizability of a schedule S, with a maximal date by an STPN $\mathcal{N}$ (that guarantees time progress):

First, unfold $\mathcal{N}$ up to a depth K computed according to the maximal date for the schedule; find the set E of all processes of $\mathcal{U}_K$ that embed S; for each symbolic process $S\mathrm{p} \in E$, compute the part of the transient tree that corresponds to executions of $S\mathrm{p}$ in $\mathcal{N}$; if the schedule imposes precise dates, include in classes a variable x_{age} that remembers the time elapsed since the beginning of an execution, and add the constraint $d(n) - \alpha \leq x_{age} \leq d(n) + \alpha$ when firing a transition implementing some node n of the schedule; then, check the probability of each path ending on a node where all transitions of $S\mathrm{p}$ have been executed; if one finds a path ρ in the transient tree with a complete execution of $S\mathrm{p}$ and nonnull probability $\mathbb{P}_\rho$, then $\mathbb{P}(S\mathrm{p} \wedge \mathrm{Sol}(\Phi_{\psi,S,d\pm\alpha})) \geq \mathbb{P}_\rho > 0$ and the algorithm can stop and return a positive answer.

6.7 Use Case: realizability in a metro network

Is this section, we develop a complete case study: we consider realizability of a particular schedule by a fleet of trains in a metro network. We show that one can depict a metro network and a simple shunting mechanism with stochastic time Petri nets, decide whether a particular schedule can be realized, and compute the probability of its realization.

Urban train systems are usually composed of closed imbricated loops, and the example developed below is a network of this kind, that is representative of the architecture of many metro lines. In metro networks, trains travel at predetermined speed profiles following a predetermined itinerary. They move from a station to another following a track. A network is hence not just a simple cycle: it contains forks, junctions, etc. Such complex topologies can easily be captured by the flow relations of a Petri net. Consider for instance the example network of Figure 6.9. This network is composed of 7 stations $sA, sB, sC, sD, sE, sF, sG$ and bidirectional tracks. Each station hence has two platforms, one for each direction, denoted respectively X and $\overline{X}$ for station sX. A train in the network can be scheduled to serve repeatedly platforms along trip $A.B.C.D.E.F.G.\bar{G}.\bar{F}.\bar{E}.\bar{D}.\bar{C}.\bar{B}.\bar{A}$, or follow smaller loop trips $A.B.C.D.\bar{D}.\bar{C}.\bar{B}.\bar{A}$ and $D.E.F.G.\bar{G}.\bar{F}.\bar{E}.\bar{D}$. This situation is a frequent one in long circular metro lines connecting suburbs and city centers. Indeed, in the mornings and evenings, it is more important to bring commuters from their home to the city center than providing long trips along the whole network.

We consider a setting where trains are isolated from one another via a so-called fixed-block policy: track portions are reserved for one particular train. This prevents trains from being too close to one another. Though this is not the only nor the most efficient way to avoid collision, this mechanism is used in real systems [1]. With such a fixed block policy, one can address a train network at block level (i.e., attach a place to each portion of the network that can be entered by at most one train), and model the effect of signaling systems that avoid collision with a blocking semantics of nets.

[1] `https://en.wikipedia.org/wiki/Railway_signalling#Fixed_block`

The Petri net at the bottom part of Figure 6.9 can represent this network: places are used to represent stations or tracks between stations, and transitions model a possible move from a part of the network to a consecutive one. The flow relation of this net is almost isomorphic to the original network.

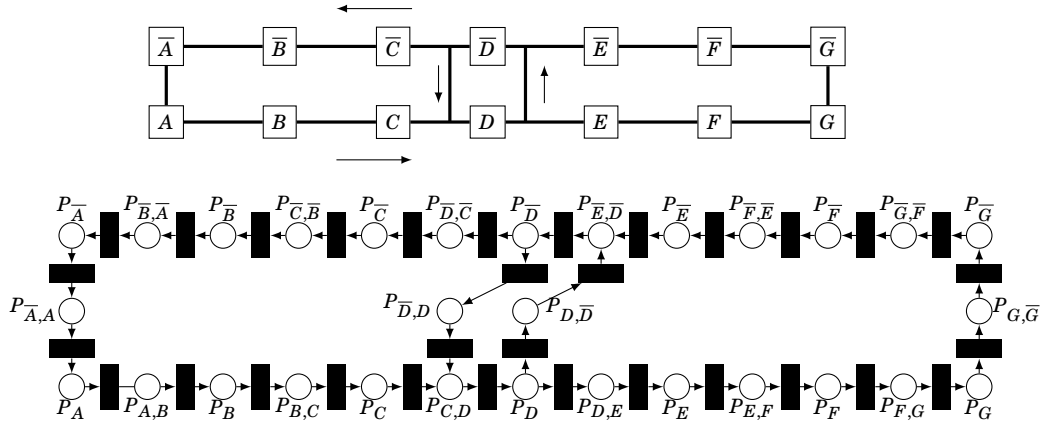


Figure 6.9: Modeling trains flows in a network with Petri nets

We will use the formalisms and the definition of realizability to give an answer to the following questions:

- Given a particular schedule, is it realizable from a given configuration of the net? Being able to answer this question allows to check whether the accumulated delays of trains force rescheduling, especially to organize trains passage at track junctions.
- From a given configuration of the network, what is the probability that all trains in a fleet complete their trips within less than 39 minutes ?

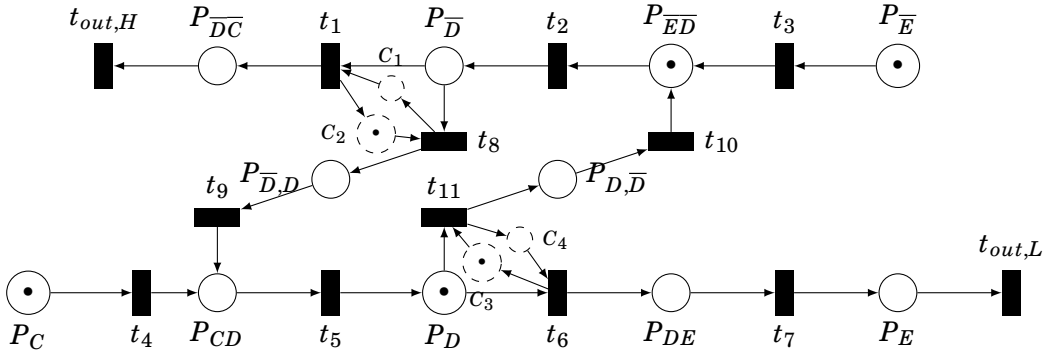


Figure 6.10: A zoom on the central part of the network

Consider the example of Figure 6.10. This net provides more details on the central part of the network of Figure 6.9. The network is decomposed into an upper and lower part. In the upper part, trains circulate from right to left, with places $P_{\overline{DC}}$, $P_{\overline{ED}}$ symbolizing track portions respectively between stations D,C and E,D in this direction, and two places $P_{\overline{E}}$ and $P_{\overline{D}}$ symbolizing platforms at stations E and D. Transitions t_1,t_2,t_3 symbolize respectively a departure from station $\overline{D}$, an arrival at station $\overline{D}$, and a departure from station $\overline{E}$. Trains leave this central part via transition $t_{out,H}$.

Similarly, the lower part of the net of Figure 6.10 represents a part of the network where trains circulate from left to right. It contains places P_{CD}, P_{DE} symbolizing track portion

between stations C,D and D,E. Places P_C,P_D,P_E symbolize respectively platforms at stations C,D,E. Transitions t_4,t_5,t_6,t_7 symbolize respectively departure from C, arrival in D, departure from D, and arrival in E. Trains leave this central part via transition $t_{out,L}$.

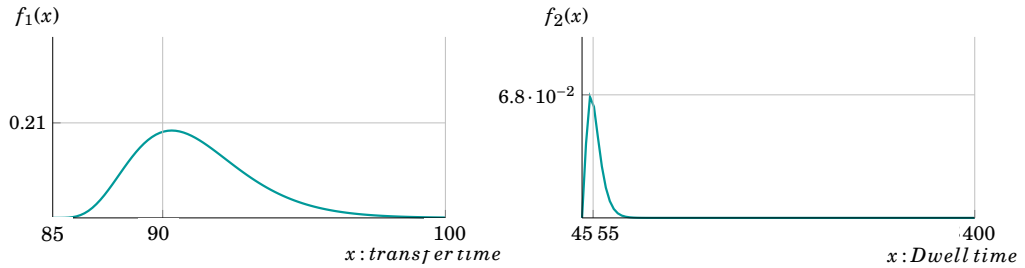
Transitions t_8,t_9,t_{10},t_{11} and places $P_{\overline{DD}},P_{D\overline{D}}$ are used to allow trains to move from the upper part to the lower part, which is needed to perform small loop trips. Last, places C_1,C_2 implement a flip-flop shunting mechanism to direct one train over two leaving platform $\overline{D}$ towards platform D, and the other towards platform $\overline{C}$. Places C_3,C_4 play the same role to direct trains leaving platform D towards $\overline{D}$ or E.

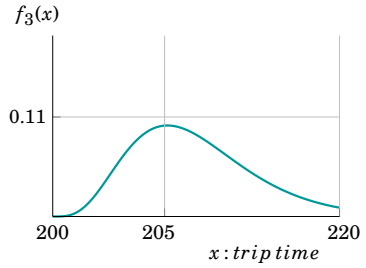
With a perfect timing of trains, this simple shunting mechanism suffices to implement two crossing small loops in the network. Consider again the network of Figure 6.10. There are four trains, represented by tokens. We will call $train_1$ the train symbolized by a token in place $P_{\overline{ED}}$, $train_2$ the train symbolized by a token in place $P_{\overline{E}}$, $train_3$ the train symbolized by token in place P_D and $train_4$ the train represented by token in place P_C.

Let us now associate durations and distributions to transitions. For the sake of simplicity, we consider identical distributions for dwell, running times and transfer from the upper to the lower part of the net. We consider interstation distances of 2 kilometers, and trains running at an average commercial speed of 35 km/h. That is, the sojourn time of a token in places $P_{\overline{DC}}$ $P_{\overline{ED}}$, P_{CD} and P_{DE} should be defined by a distribution with maximal probability around 205 seconds, with more probable delays than advance. We furthermore consider that the distance needed to go from D to the track portion $\overline{ED}$ is 0.5 km, but is performed at a speed of 20 km/h (similarly for moves from $\overline{D}$ to track CD). Within this setting, sojourn time in places $P_{\overline{DD}}$ and $P_{D\overline{D}}$ should be a distribution centered around 90 seconds. Last, we associate a dwell time to every station. We choose arbitrarily a value of 50 seconds for the most probable dwell time in the whole network. It is very frequent that trains get late at stations due to passengers misbehavior. However, we consider that dwell time cannot exceed 400 seconds. We hence associate to transitions t_9,t_{10} (transfer) the distribution $f_1 = 1.477 \cdot 10^{-2} \cdot (x-45)^5 \cdot e^{-1.1 \cdot (x-45)}$, defined only on interval $[85,100]$. We associate to transitions $t_2,t_5,t_7,t_{out,h}$ (trips) distribution $f_3 = 1.3413 \cdot 10^{-3} \cdot (x-200)^4 . e^{-0.5 \cdot (x-200)}$ defined only on interval $[200,220]$. We last associate distribution $f_2 = 7.8125 \cdot 10^{-3} \cdot (x-45)^2 \cdot e^{-0.25(x-45)}$ defined on interval $[45,400]$ to all other dwell transitions. The intervals associated to transitions, and the shape of distributions are depicted on Figure 6.11.

Ideally, the expected trajectory ot trains 1 and 2 go through stations $\overline{D}$, D and E, in this order. The trajectories of trains 3 and 4 go through stations D, $\overline{D}$ and $\overline{C}$. This can be represented by the schedule of Figure 6.12. As we only consider departures from platforms, we associate to nodes of the schedule labels of the form dX, where X is the name of a platform. We can now relate departures from stations in the schedule and concrete events in the net on Figure 6.10 using a realization function r, such that $r(d\overline{D}) = \{t_1,t_8\}$, $r(d\overline{E}) = \{t_3\}$, $r(dC) = \{t_4\}$, $r(dD) = \{t_6,t_{11}\}$, $r(dE) = \{t_{out,L}\}$. One can notice that there are two possible ways to implement a departure from D or $\overline{D}$.

Obviously, using the simple flip-flop shunting mechanism explained above, such a schedule can be realized only if the order of trains in place $P_{\overline{D}}$ is train 1, train 3, train 2, train 4, and if the passage order in place P_D is train 3, train 1, train 4, train 2. Let us assume that we start from an initial configration where trains 1 and 3 have a remaining trip





transition	$I(t_i)$	$\psi^{-1}(t_i)$	$f(x)$
t_1	$[45,400]$	$d\overline{D}$	f_2
t_2	$[200,220]$	–	f_3
t_3	$[45,400)$	$d\overline{E}$	f_2
t_4	$[45,400]$	dC	f_2
t_5	$[200,220]$	–	f_3
t_6	$[45,400]$	dD	f_2
t_7	$[200,220]$	–	f_3
t_8	$[45,400]$	$d\overline{D}$	f_2
t_9	$[85,100]$	–	f_1
t_{10}	$[85,100]$	–	f_1
t_{11}	$[45,400]$	dD	f_2
$t_{out,H}$	$[45,400]$	–	f_2
$t_{out,L}$	$[45,400]$	dE	f_2

Figure 6.11: Distributions

duration of 10 seconds, that trains 2 is delayed and cannot leave $P_{\overline{E}}$ before 320 seconds, that train 4 is delayed and cannot leave P_C before 300 seconds. This initial configuration of the network can be easily encoded by attaching adequate times to fire to enabled transitions t_2, t_3, t_{11}, t_4. Considering the schedules, one can safely add the constraint that all trips are performed after 2300 time units. As the interval attached to transitions all have lower bounds greater than 45, and as the net has 13 transitions, all timed processes of the net in Figure 6.10 embedding the schedule of Figure 6.12 in less than 2300 time units appear in an unfolding $\mathscr{U}_K$ of depth at most $K = \lceil \frac{2300}{45} \rceil \cdot 13^2 = 8788$. Note however that if trains behave as expected, all trains modeled in the example of Figure 6.10 will eventually leave this part of the network after a finite time, and hence unfolding should stop much earlier due to lack of appendable transitions. Yet, even without this a priori information, as dwells and running consume time, this allows to represent the behavior of the network with an unfolding of bounded depth. Then, from the initial configuration, there exists time processes satisfying all time constraints due to dwell and running times. Figure 6.13 is an example of such process. As usual, conditions are represented by circles and events by squares. Due to lack of space, we do not give all constraints attached to this process. They will contain constraints on event variables of the form:

$$
\begin{aligned}
& doe(t_2^1) = 0 \\
& dof(t_2^1) = doe(t_2^1) + 10 \\
& dof(t_2^1) \leq \theta(t_2^1) \\
& .... \\
& \left[\theta(t_3^1), \theta(t_2^3)\right] \cap \left[\theta(t_{10}^1), \theta(t_2^2)\right] = \emptyset \\
& ....
\end{aligned}
$$

We indicate with red numbers associated with events a possible dating function d. One

can verify that for any ordered pair of events $e \leq e'$, we have $d(e) < d(e')$, that sojourn times in places are compatible with dwell and running times, and that for any pair of conditions P_X^i, P_X^j with $j \neq i$ we have $\left[d\left({}^\bullet(P_X^i)\right), d\left((P_X^i)^\bullet\right)\right] \cap \left[d\left({}^\bullet(P_X^j)\right), d\left((P_X^j)^\bullet\right)\right] = \emptyset$.

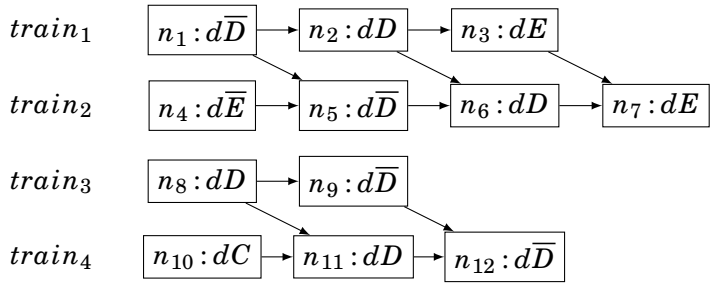


Figure 6.12: A possible schedule for trains departures from initial configuration in Figure 6.10

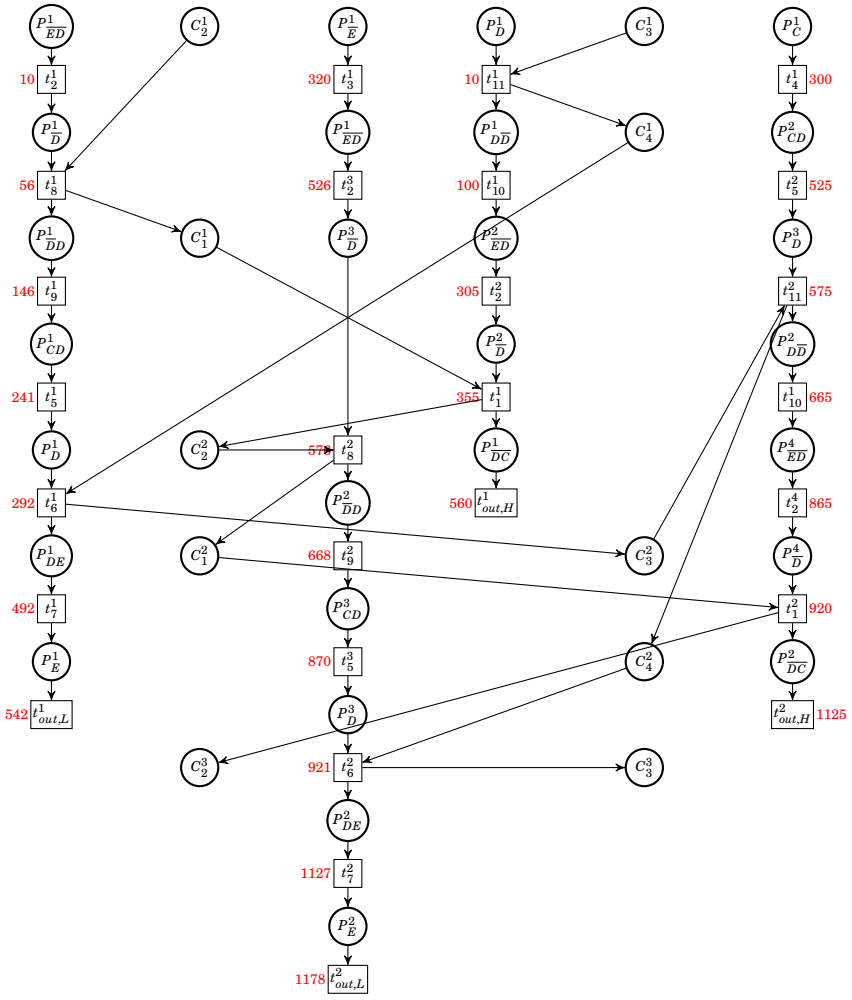


Figure 6.13: A witness for realizability of schedule in Figure 6.12

The time process of Figure 6.13 shows a possible realization of the schedule in Figure 6.12. As we are dealing with continuous probabilities the probability of realizing exactly this process with the specified dates is 0. However, one can easily notice that all dates indicated on this process can be shifted by several time units. Indeed, as dates are not

discrete values, there is an infinite number of time processes with the same conditions and events but with different timing functions. This defines a class $\mathscr{C}$ of time processes realizing our schedule with a non-null probability $\mathbb{P}(\mathscr{C})$. Note that the probability of realizing the schedule is greater than the probability $\mathbb{P}(\mathscr{C})$ of executing a time process from this class, as there could be more than one way of realizing the schedule. The value $\mathbb{P}(\mathscr{C})$ can be computed by construction of a transient execution tree, starting from the initial configuration, and in which nodes are of the form (M_i, C_i, D_i), where M_i is a marking, C_i represents possible values of TTFs attached to enabled transitions, and D_i the probability distribution with positive values on the domain C_i defined this way.

An execution of a time process from $\mathscr{C}$ necessarily starts from the initial configuration given above, i.e, from node $(M_0, C_0, D_0, \text{BLK}_0, \text{URG}_0)$, with $M_0 = \{P_{\overline{E}}, P_{\overline{ED}}, P_C, P_D\}$, $\text{BLK}_0 = \emptyset, \text{URG}_0 = \emptyset$. For each enabled transition t_i, we define by x_i the variable symbolizing possible values of TTF (remaining time to fire) of t_i. We hence have $C_0 = \{x_2 = 10; x_3 = 320; x_4 = 300; x_{11} = 10\}$, i.e. C_0 is a point in $\mathbb{R}^4_{\geq 0}$. The distribution D_0 associates probability 1 to the single point in C_0. From this situation, two events can occur: either t_2 fires after 10 time units, or t_{11} fires after 10 time units. As both events are enabled at the same date, the probability to fire each one is defined according to weights attached to transitions. If we assume that t_2, t_{11} have the same weight, so the probability to fire t_2 from (M_0, C_0, D_0) is 0.5.

Let us consider the effect of firing t_2 at date 10: it affects marking, and remaining TTFs. After firing t_2 the possible configurations of the net are encoded by a node $(M_1, C_1, D_1, \text{BLK}_1, \text{URG}_1)$ where $M_1 = \{P_{\overline{E}}, P_{\overline{D}}, P_C, P_D\}$, $\text{BLK}_0 = \emptyset, \text{URG}_1 = \{t_1 1\}$. Upon firing of t_2, transition t_8 becomes enabled, and a new time to fire represented by variable x_8 is sampled. Hence the possible values for TTFs are depicted by the constraints in class $C_1 = \{x_3 = 310; x_4 = 290; 45 \leq x_8 \leq 400\}$. The distribution D_1 attached to this class is $D_1(x_3, x_4, x_8) = f_2(x_8)$ if $(x_3, x_4, x_8, x_{11}) \in C_1$, and 0 otherwise.

From this class, transition t_{11} is urgent and has to fire immediately, with probability 1 yielding a new marking, and sampling of a new value for newly enabled transition t_{10}, and a new node $(M_2, C_2, D_2, \text{BLK}_2, \text{URG}_2)$ with $M_2 = \{P_{\overline{E}}, P_{\overline{D}}, P_C, P_{D\overline{D}}\}$, $\text{BLK}_2 = \emptyset, \text{URG}_2 = \emptyset$. A new value for TTF of transition t_{10} is sampled, yielding class $C_2 = \{x_3 = 310; x_4 = 290; 45 \leq x_8 \leq 400; 85 \leq x_{10} \leq 100\}$, and distribution D_2 such that $D_2(x_3, x_4, x_8, x_{10}\} = f_2(x_8).f_1(x_{10})$ if $(x_3, x_4, x_8, x_{10}) \in C_2$, and 0 otherwise.

The next classes that can appear are more complex, and computing them is more involved. Assume that transition t_8 fires from a configuration of class C_2. This means in particular that t_8 is the transition with the smallest TTF, that is x_8 is smaller than x_3, x_4, x_{10}. One can easily check that some values for x_3, x_4, x_8, x_{10} satisfy $C'_2 = C_2 \cup \{x_8 \leq x_3; x_8 \leq x_4; x_8 \leq x_{10}\}$. This additional constraint has to be considered, both to compute the probability of firing t_8 and to compute the new class reached after firing t_8. The class obtained after t_8 is $(M_3, C_3, D_3, \text{BLK}_3, \text{URG}_3)$ and is computed as follows. First $M_3 = \{P_{\overline{E}}, P_{\overline{D}D}, P_C, P_{D\overline{D}}\}$. Then, the new set of constraints is obtained using the standard construction of state classes:

- start from $C'_2 = C_2 \cup \{x_8 \leq x_3; x_8 \leq x_4; x_8 \leq x_{10}\}$.
- do a variable substitution of the form $x_i = x_8 + x'_i$ for all variables. This substitution allows to propagate the fact that new values for x_3, x_4, x_{10} are decreased by the value of x_8.
- eliminate all variables related to disabled transitions (using the Fourier-Motzkin method). In our case, only x_8 must be eliminated.

- add inequation(s) related to newly enabled transition(s). In our case, as t_9 becomes enabled, this amounts to inserting constraint $85 \leq x_9 \leq 100$.
- rename all x'_i into x_i

The class C_3 obtained by substitution and elimination from C'_2 is $C_3 = \{x_3 - x_4 = 20; 0 \leq x_3 \leq 265; 0 \leq x_4 \leq 245; 0 \leq x_{10} \leq 55; 210 \leq x_3 - x_{10} \leq 225; 190 \leq x_4 - x_{10} \leq 205; 85 \leq x_9 \leq 100\}$.

Let us now consider distribution D_3. We can compute the probability p of firing transition t_8 from node (M_2, C_2, D_2) as the integration over all possible values of vector x_3, x_4, x_8, x_{10} over C'_2, i.e.

$$p = \int_{C'_2} D_2(x_3, x_4, x_8, x_{10}) = \int_{45}^{100} \int_{x_8}^{100} f_2(x_8) f_1(x_{10}) d_{x_{10}} d_{x_8}$$

This value can be approximated to 0.171499, i.e., firing of t_8 is not the most probable event in $(M_2, C_2, D_2, \text{BLK}_2, \text{URG}_2)$. We can now build D_3, using the procedure given by [39]. As t_8 fires before any other transition from class (M_2, C_2, D_2) we first compute a new distribution D_2^a defined over C'_2 such that $D_2^a(x_3, x_4, x_8, x_{10}) = \frac{D_2(x_3, x_4, x_8, x_{10})}{p}$. The next step is to build the distribution of probabilities once x_8 is eliminated, i.e. a distribution $D_2^b(x_3, x_4, x_{10})$ computed as $D_2^b(x_3, x_4, x_{10}) = \int_{45}^{100} D_2^a(x_3 + x_8, x_4 + x_8, x_8, x_{10} + x_8)$. In general, this integration is repeated for every variable attached to a disabled transition, but in the current case, only x_8 needs to be projected away. The last step consists in integrating the newly created variable x_9 and its distribution to D_2^b. As the value of x_9 does not depend on former values of x_3, x_4, x_{10}, we have $D_3 = D_2^b.f_1(x_9)$, defined over a domain C_3 for variables x_3, x_4, x_9, x_{10}.

We can repeat this process for all transitions that are firable from any node to build a transient execution tree. At the end of the construction, we can distinguish a set $\mathscr{PT}$ of paths of the tree that end after execution of all events appearing in the process of Figure 6.13. We can associate to each of these paths $\rho \in \mathscr{PT}$ a probability p_ρ that is the product of probabilities of each transition in ρ. Last, the probability $\mathbb{P}(\mathscr{C})$ to execute the process of Figure 6.13 is $\mathbb{P}(\mathscr{C}) = \sum_{\rho \in \mathscr{PT}} p_\rho$.

One of the key points in this technique is to integrate over variables domains. Recall that $\int_a^b \alpha \cdot x^n \cdot e^{-\lambda \cdot x} = -\alpha \cdot \lambda \int_a^b x^n \cdot \frac{-1}{\lambda} e^{-\lambda \cdot x} = -\alpha \cdot \lambda([x^n \cdot e^{-\lambda \cdot x}]_a^b - \int_a^b n x^{n-1} \cdot e^{-\lambda \cdot x})$. Hence, probabilities can be computed exactly through an iterative process.

Let us now show how to ensure that an event n represented in a schedule occur at a fixed date $d(n) \pm \alpha$. As shown in [39], it is sufficient to add to state classes a variable x_{age} initially set to 0, and that is incremented by the value of every variable x_i at every state class updated resulting from the firing of a transition t_i. In other words, in every class, variable x_{age} describes possible value for the time that has elapsed since the beginning of the execution of a process. When an event of a process representing occurrence of a node n (i.e, such that $\psi(n) = e$ which date must be $d(n)$ up to imprecision α, it suffices to add to the domains of the newly computed state class the constraint that $d(n) - \alpha \leq x_{age} \leq d(n) + \alpha$. Integration follow the same principles as before, but over domains with one additional dimension. For instance, if we impose that node n_1 in the schedule of Figure 6.12 is executed at a date 60 ± 10, is suffices to maintain variable x_{age} during construction of states classes $(M_0, C_0, D_0, \text{BLK}_0, \text{URG}_0), (M_1, C_1, D_1, \text{BLK}_1, \text{URG}_1), (M_2, C_2, D_2, \text{BLK}_2, \text{URG}_2)$ as defined above, and to impose during the construction of state class C_3 the additional constraint that $50 \leq x_{age} \leq 70$. Similarly, completing a schedule S within a maximal amount

of time Δ amounts to imposing constraint $d(n) \leq \Delta$ at every maximal node n of S. This means refining state classes reached by firing transitions representing these maximal node with constraint $x_{age} \leq \Delta$. Answering our second question on the use case then amounts to checking for every process that embeds S and every final path in the associated transient tree that constraint $x_age \leq (39 * 60)$ allows firing at least one transition reaching a final node (i.e. where the schedule is completely executed) with positive probability.

7

Simulation and Performance Evaluation

This chapter addresses performance evaluation of traffic management techniques in urban rail systems. The approach adopted for this purpose is simulation-based and computes mean values on KPIs after several simulation runs. The simulation framework implements the model of Section 5.2 of Chapter 5. This model is implemented in the SIMSTORS tool. The tools takes as input a description of a network, a reference timetables describing train services and thresholds on desired dwell and running times of trains. SIMSTORS provides an efficient simulation framework : it can play several hours of rail operation in less than a minute. This allows for simulation campaigns producing large sets of logs or randomly generated runs, and then for the calculus of KPIs from these logs. In this chapter, we demonstrate the capabilities of this simulation scheme on a case study, and show how statistics can be derived through simulation campaigns.

7.1 Introduction

In Chapter 5, we have introduced Key Performance Indicators (KPIs) and regulation techniques. The simplest of these techniques generally try to stick to a prescribed timetable but complex proprietary algorithms are also in use. One can however notice that there is no consensual mechanism considered as the best, as efficiency of such techniques depends on many contextual factors such as line topologies, frequency of delays, or even passenger civic-mindedness. Considering traffic management problems and evaluating the performance of the to-be-used techniques on a line or network at early design stages has several advantages. First, it allows to decide which technique is adapted to particularities (e.g., interstation length, maximal train commercial speeds, number of trains) of the line under construction. Second, it allows to build timetables and to estimate achievable performance.

Several tools have been used to evaluate performance of mainline rail systems. Follow-

ing the classification in [54], one can define these tools as macroscopic or microscopic simulation tools. Macroscopic approaches abstract away details (e.g., fine modeling of train acceleration and deceleration adherence to tracks...) and do not to consider trains particularities for simulation; they usually lead to optimistic results. An example of such macroscopic models is the NEMO tool [41]. This tool uses abstract network graphs to compute timetables and detect possible bottlenecks. Microscopic approaches consider many parameters of trains, of networks, and of their environments such as weather conditions, passenger flows... Usually, these approaches consider how trains influence one another at runtime. They use discrete-time synchronous techniques, i.e., repeatedly evaluate evolution of a network during user-defined time steps (e.g., one second). OpenTrack [54] is an example of such simulation frameworks. Synchronous simulation is time consuming, and many steps simulated by the tools are simply useless, as no interaction between trains (forcing one of them to brake, for instance) nor change in trains behavior (excepted for their positions) occurs during most of time steps. OpenTrack and NEMO, as well as commercial software such as RailSys [58], target mainline systems, where delays between departures and arrivals are rather long, and where small local disturbances have little influence on service performance. Challenges for these models are to design timetables, that are quite stable, and in case of failure in a network, find alternative paths for trains. Computation of best alternative routes can take a few minutes without affecting too much traffic. In urban networks, paradigms change: distances between trains are small, minor disturbances may affect service quality, and advice has to be computed as fast as possible to be of use. Hence, corrective mechanisms are quite reactive, and the computed solutions to recover from a delay are applied as soon as possible. Models such as those proposed in the SimMETRO tool [44] address performance of metro systems in a microscopic (and stochastic) setting.

In this chapter, we describe a macroscopic performance evaluation scheme for traffic management algorithms in urban rail systems, that can be used at early design stages.

We use as a model for Urban Train Systems the blocking STPNs proposed in Chapter 5. This model is abstract enough to allow efficient simulation (many characteristics of trains, speed profiles, tracks and so on are abstracted away), yet accurate enough to derive useful performance measures. We show that KPIs can be easily evaluated from our model, and demonstrate its practical interest on a real case study, namely line 1 on Santiago's metro.

7.2 Modeling

The design of our simulation framework, is modular: we decompose the system into 3 distinct parts that communicate:

1. **Network and trains** physical parts of a rail system (trains, tracks and signaling), movements of trains (departures and arrivals), nominal operation times (dwell and running times), and disturbances (advance and delay) are represented via an STPN model equipped with elementary semantics;
2. **timetable** Our simulation algorithm uses a *reference timetable* depicting ideal departure and arrival dates for trains, and an active timetable that is updated online at execution time to handle primary delays and the changes brought by regulation to cope with these disturbances.

3. **Regulation** The last part of the simulator is composed of a traffic management algorithm (control), implemented as a function that, when triggered, computes new dates and updates the online timetable.

The whole system is simulated according to the the semantics of STPNs with blocking semantics described in Section 5.2.4. Fig. 7.1 provides an overview of the simulation framework. The STPN is at the bottom of the Figure; control is the dotted box that sends orders to the STPN; and events are represented by a timetable, and are recorded in a log at the end of each simulation run.

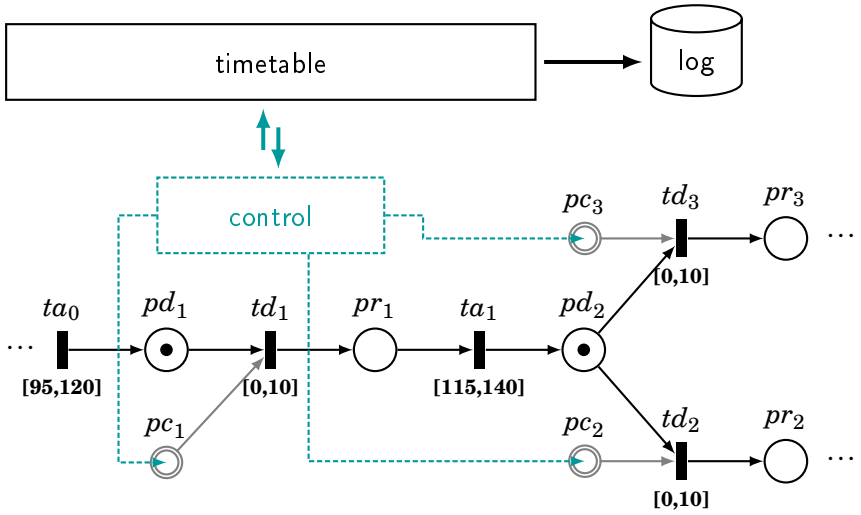
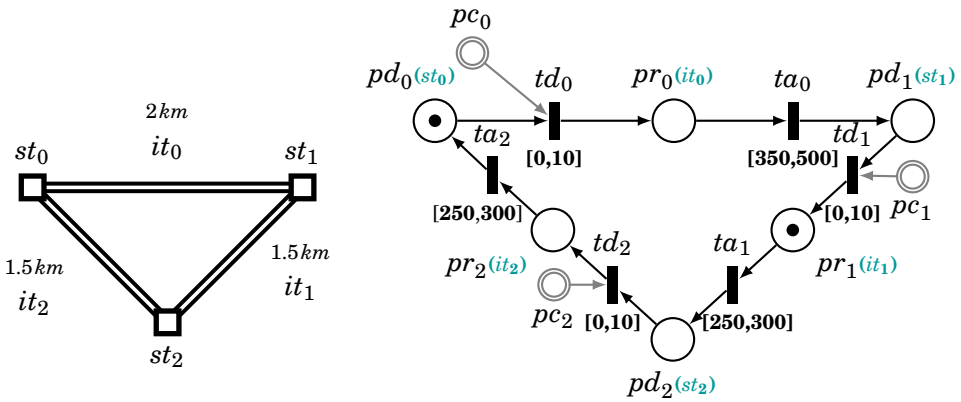


Figure 7.1: The SIMSTORS simulation framework

STPNs with blocking semantics are abstractions of systems with a fixed block safety policy. This safety policy is implemented on existing metro lines, and in particular in the use case studied in this chapter, namely line 1 of Santiago's metro. The same framework can be used to study performances of systems with a moving block policy by replacing the STPN component with a trajectory Petri net (cf. Section 5.3).

We recall that STPN models are equipped with *control places* (depicted by doubly circled places in Figure 7.1), that are used to guarantee that trains leave a station only after a departure order is explicitly given. Control places hence model departure orders. A control place is created for each station, and during execution time, a token in a control place means that the departure from this station is allowed. As we use the blocking semantics of section 5.2.4, departures never occur earlier than planned. However, one can easily adapt this semantics to allow slight advances in the execution of the timetable part, to allow advances. As previously stated, components of the simulation framework communicate with each other. In the initial state of the system, the STPN component comes with empty control places. These places are filled when the online timetable component indicates that the current date is equal to the planned date of a departure.

Let us consider the toy ring network of Figure 7.2. The left hand side subfigure depicts a topology for a network with 3 stations st_0, st_1 and st_2, and 3 interstations it_0, it_1 and it_2, respectively of length $2kms$, $1.5kms$, and $1.5kms$. Train movements are performed clockwise, i.e., $st_0 \rightarrow st_1 \rightarrow st_2 \rightarrow st_0 \cdots$ To simplify the example, we consider that each interstation is a single block. The right part of the figure represents a STPN modeling behaviors of trains in this network. One can immediately notice that the graph composed



(a) (b)

Figure 7.2: STPN modeling of a simple ring topology with two trains

of places, transitions and flow relation of the STPN is a simple morphism of the graph depicting the topology of the net corresponding synthesized: stations are symbolized by places, and interstations replaced by a pattern with two transitions (symbolizing departure and arrival) and two places (symbolizing track from a station to the next one, and departure orders). A similar mapping can be performed for any kind of network topology. If one want to have more than one block between two stations, the simple Petri net pattern is replaced by a longer patter with more places and more transitions. In the STPN model of a URS, the set of places can be partitioned into a set of places representing location of trains during dwell operations (pd_0,$pd_1\ldots$), and during running operations (pr_0,$pr_1\ldots$). One can also add places representing safety mechanisms (ps_0,$ps_1\ldots$), for instance to prevent two trains entering a share track portion. Such situations can be imposed in a network, beyond the blocking semantics, for instance when sharp curves prevent trains crossing in opposite directions. Last, the set of places contain a set of control places (pc_0,$pc_1\ldots$ in our example, represented as double-lined circles). Transitions can be partitioned into two subsets: transitions for departures (td_0,$td_1\ldots$) and transitions arrivals (ta_0,$ta_1\ldots$). We associate to each transition an interval and a CDF (not shown on the figure) defined on this interval. intervals represent the possible ranges of dwell and running time, and also give means to introduce random delays in the system. The last ingredient to integrate to the model is a representation of trains. In STPNs with a blocking semantics, trains are represented by a token located in a dwell or running place. In our example, a trains is stopped at stations st_0, and another train is on its way from station st_1 to st_2.

Now, STPNs only describe the dynamics of trains, i.e., how they move from one track portion to another, and the time needed to move from one part of the network to another (with possible random delays). As already mentioned, delays are recovered using traffic management techniques that should, hence, be considered when evaluating the overall dynamics and performance on an urban rail network. Indeed, two different techniques will not lead to the same performance. This justifies the modularity in the design of our model that separates the traffic management algorithm from the rest of the system. We can hence build once for all an STPN for a given topology, and use the same model with different regulation algorithms to measure the KPIs achieved with a particular regulation.

Our simulation framework follows the semantics given in Chapter 5. So, the STPN is not executed in isolation: train departures await orders from the control part, that are generated only at dates planned in the timetable. In the same vein, involved regulation techniques can change the trip of trains by sending departure orders to transitions symbolizing departure on different track portions. However, in this chapter, we will mainly simulate regulation techniques that change events dates, but not their ordering. In the example of Figure 7.2, let us assume that the regulation used is a schedule policy (explained in Section 3.3.2 that tries to stick to a articular reference timetable. transition ta_1 is enabled by the token in place pr_1. Let us suppose that we are at date $d_0 = 0$, and that the sampled time-to-fire value following this enabling is $\tau_1 = 280 \in [250, 300]$. We recall that the distribution associated with transition ta_1 gives probabilities for the running time on interstation it_1 (from st_1 to st_2), distributed around a nominal value. Let us assume that the nominal value for the running time in interstation it_1 is 275. This means that the arrival at station st_2 represented by firing of transition ta_1 will occur 5 time units later than expected. In order to recover from this delay as early as possible, our chosen traffic management technique will simply adapt the date next departure from station st_2: it will attempt at forcing it to occur at a date earlier by 5 $(280 - 275)$ time units than the nominal departure date. If the nominal dwell time is, e.g., 30 time units, then departure from st_2 should normally occur at date 310 $(280 + 30)$, if regulation does nothing. However, if the regulation adapts slightly the departure dates in the timetable, a departure can occur at date 305 $(275 + 30)$. Following adaptation of the timetable, a token will be inserted in the control plce of the departure transition at this date 305, allowing to recover from the primary delay. Note that it is still possible that this date of departure will not be respected if, e.g., the sampled dwell time imposes a new additional delay. Notice that with a schedule policy, one can only recover from delays at stations, nd as a minimal dwell time is always imposed, the amount of time recovered at each station is limited.

Implemented this way, our simulator relies on an abstract representation of train moves but integrates **real** traffic management mechanisms. Even if the overall system built this way is too complex to be formally analyzed (we recall that regulation algorithms can be any type of code transforming a timetable), each module of our simulation framework is simple enough to be designed separately. Equipped with a formal semantics (where the regulation algorithm Is simply defined as a deterministic function), our framework is well adapted to simulation. We have paid a particular attention to efficiency during the application of semantics rules: to check that a transition is enabled, for instance, a well adapted data structure needs only to consider the contents of places in the preset of a transition. Similarly, firing a transition only changes the enabling of transitions which consume tokens from the newly marked places, etc.

We have used this simulation framework to evaluate the performance of a traffic management technique on line 1 of Santiago's metro. This line is a complex ring topology with intertwined loops connecting 24 stations (cf. Figure 7.3). The STPN built for this line is a net with a total of 645 places (all types included) and 294 transitions. The model contains depots and turn back zones in addition to stations and their interconnections. With this STPN, we can simulate 4 hours of operation of this line with 50 trains and random disturbances, in approximately 19 seconds, on a computer with a CPU running at 2.60 GHz.

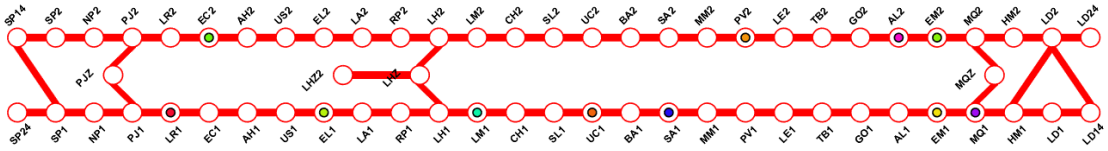


Figure 7.3: Line 1 of Santiago's metro

7.3 Simulation and results

We have simulated the first 4 hours of operation for line 1 of Santiago's metro, with a total of 50 trains operating on the line. Traffic is not immediately maximal but increases progressively as trains are inserted in the network. The system was equipped with a basic regulation algorithm that aims to stay as close as possible to a precomputed ideal timetable TT^{id}. The regulation adapts dwell times to recover from unexpected delays, and maintains a feasible timetable that associates to departures and arrivals their earliest possible occurrence date. We have repeated 100 times a simulation of 4 hours of operation, recording each time departure and arrival dates at all stations. During each simulation, dwell and running times for each event were randomly sampled from their respective distributions. During this experimentation, the distributions attached to transitions were discretizations of asymmetric bell shaped curves (close to a discretization of expolynomial functions).

At the end of the simulation campaign, the obtained data were a log of 100 simulated runs, i.e. successions of departure and arrival dates for all steps of realized trips. Overall, the simulation campaign took around 1 hour. Once obtained, the logs allow for computation of statistics and of KPIs. The KPI we have considered was the mean deviation w.r.t. desired departure headways. This KPI measures to what extent a reference timetable was properly realized.

Timetable deviations

Figure 7.4 depicts the mean deviations computed for each individual simulation. Abscissa indicate the simulation number (ranging from 1 to 100). The different curves on the picture represent the mean deviation w.r.t. to the ideal timetable TT^{id} at each station (1 curve per station). The dark curve represents the data for one highlighted station, namely Pajaritos, in running direction 1. We recall that we slightly abuse the term *station*, as for each physical location of line 1, we have a station number for each running direction. (There are two possible directions: direction 1 from station San Pablo to Los Dominicos, and direction 2, the converse way.) From these recorded mean deviations w.r.t. timetable TT^{id}, one can observe the randomness in the simulation, as for each run, we obtain different results.

Let n be the number of simulations performed during a campaign (in our case $n = 100$), and let r_j with $j = 1,2,\ldots,n$ denote the j^{th} simulation run. Let m be the number of stations, and let us denote by st_k the k^{th} station on the line, with $k = 1,2,\ldots,m$. Events occurring at a given station st_k in a run r_j are denoted $e_{i,j,k}$ with $i = 1,2,\ldots,q_k$. Note that, during our simulation campaign, the number of events per station was the same from a run to another.

Given an event $e_{i,j,k}$ (in run r_j, at station st_k), we denote by $d_{i,j,k}$ the *reference date* for

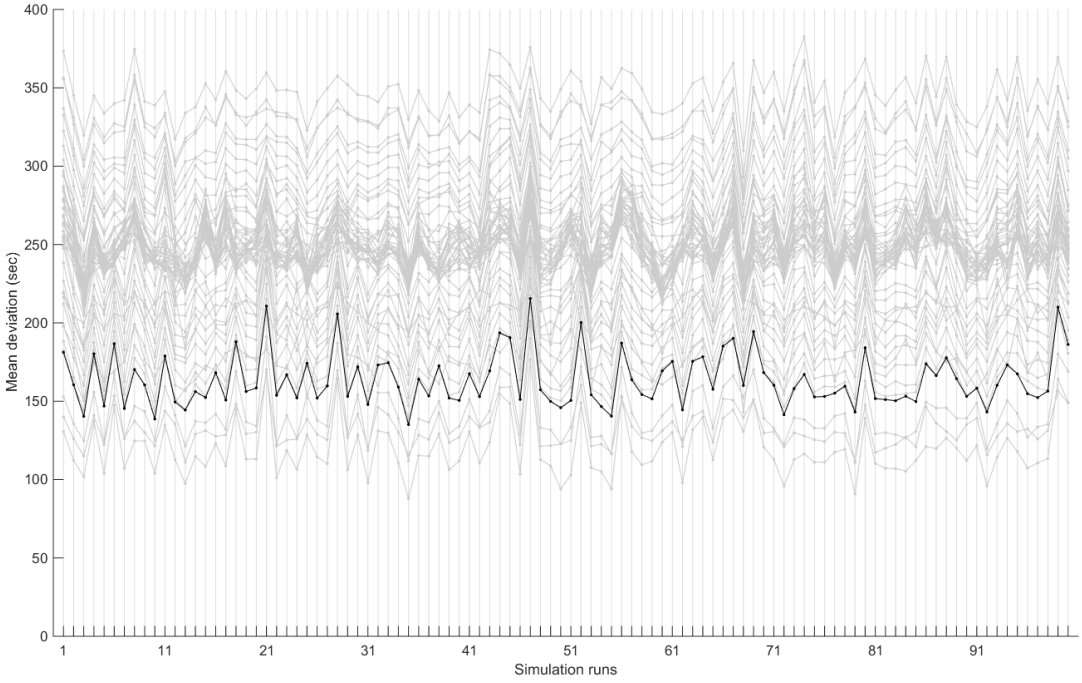


Figure 7.4: Mean deviations from reference timetable for $n = 100$ simulation runs

event $e_{i,j,k}$, found in the ideal timetable TT^{id}. It is is the ideal occurrence date at which the event occurs if it is not delayed. Note that as $d_{i,j,k}$ is the same for all runs, we can simply write it as $d_{i,k}$. On the other hand, the *effective occurrence* of event $e_{i,j,k}$ is denoted by $\dot{d}_{i,j,k}$. The difference between these two dates for the same event is called the *deviation* (from the reference timetable) and is denoted by $\delta_{i,j,k} \triangleq \dot{d}_{i,j,k} - d_{i,k}$.

Fig. 7.5 shows data collected during a single run of our simulation. It is a graph in which several curves are superimposed. Each curve represents the evolution of deviations at one station (in one direction of movement). Here, the curve associated with station Pajaritos in direction 1 is highlighted with a dark line. Curves are superimposed to show the general tendency of the evolution of deviations. Abscissae give occurrence dates $\dot{d}_{1,1,k}, \dot{d}_{2,1,k}, \ldots, \dot{d}_{q_k,1,k}$, and ordinates give deviations $\delta_{1,1,k}, \delta_{2,1,k}, \ldots, \delta_{q_k,1,k}$. It might seem surprising that deviations grow but this is due to the chosen parameters for the simulation: we have deliberately selected high values of disturbances to be able to observe the impact of regulation. One can see that, in the beginning of the simulation, regulation is able to recover, more or less, from the disturbances but, as time progresses, the system becomes unstable. This is due to the fact that more and more trains are inserted into the network. As a consequence, it becomes harder for regulation algorithms to recover from consequent delays, and bunching phenomena appear. That is, when a train is delayed for too long, the distance between this train and its successor decreases. If this phenomenon lasts for a too long time the whole fleet is is grouped in a small area of the network, and each train is separated from its predecessor by a distance close to the safety headway. Such configuration is undesirable as, first, the reference timetable is not respected, and second, quality of provided service is highly degrades. Indeed, in buching situations, as passengers have been waiting for a long time before a train arrives, they all board the first trains. As a result, bunching leads to fleets composed of one or two heavily crowded vehicles followed by almost empty ones.

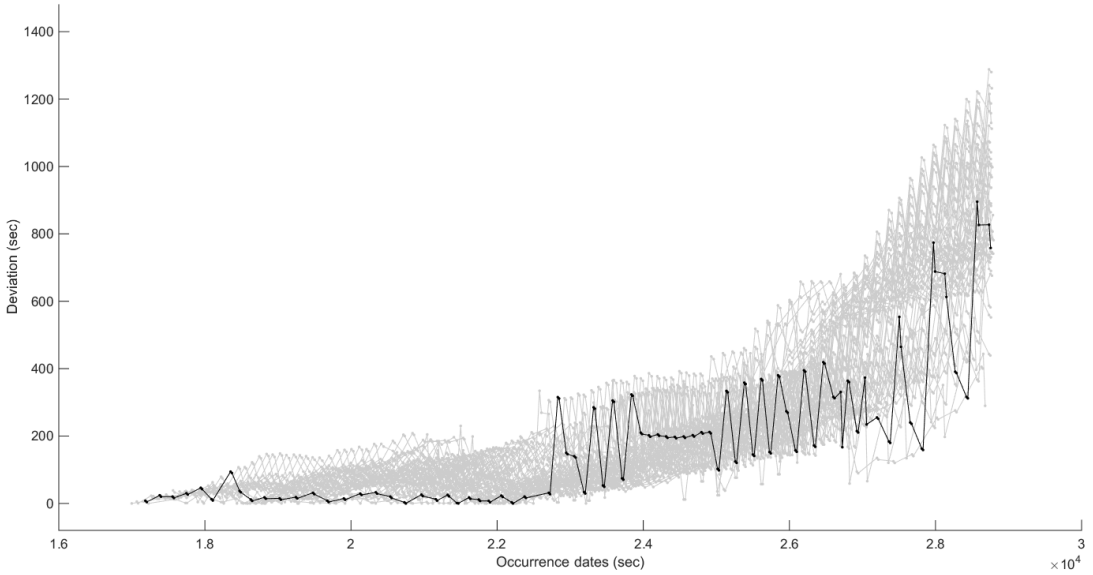


Figure 7.5: Evolution of deviations w.r.t. ref. timetable for one simulation run at each station

Headways

Now, instead of reasoning in terms of deviations w.r.t. occurrence dates of arrivals or departures, one can also consider *headways*, as they give a better measure of traffic regularity. For headways to be relevant, they have to be measured only between events of the same type (i.e., departures or arrivals). We will, thus, respectively write $e^d_{i,j,k}$ and $e^a_{i,j,k}$ to denote the i^{th} departure and arrival at station st_k in run r_j. Also, q^d_k and q^a_k denote the total number of departures and arrivals at station st_k (no mention of index j as there is no need to differentiate between runs). In this context, a headway is the difference between dates of two consecutive events of the same type. $h^d_{i,k} \triangleq d(e^d_{i+1,k}) - d(e^d_{i,k})$ is the *reference headway* at the $(i+1)^{\text{th}}$ departure from station st_k, and $h^a_{i,k} \triangleq d(e^a_{i+1,k}) - d(e^a_{i,k})$ is the reference arrival headway. We denote by $\dot{h}^d_{i,j,k} \triangleq \dot{d}(e^d_{i+1,j,k}) - \dot{d}(e^d_{i,j,k})$ the *effective headway* at departure $i+1$ in run r_j at station st_k, and $\dot{h}^a_{i,j,k} \triangleq \dot{d}(e^a_{i+1,j,k}) - \dot{d}(e^a_{i,j,k})$ the effective headway at arrival. We can then define $\bar{h}^d_k \triangleq \sum_{i=1}^{q^d_k-1} h^d_{i,k} / (q^d_k - 1)$ and $\bar{h}^a_k \triangleq \sum_{i=1}^{q^a_k-1} h^a_{i,k} / (q^a_k - 1)$ as the *mean reference headways* for departures and arrivals at station s_k. Also, $\widetilde{h}^d_{j,k} \triangleq \sum_{ix=1}^{q^d_k-1} \dot{h}^d_{i,j,k} / (q^d_k - 1)$ and $\widetilde{h}^a_{j,k} \triangleq \sum_{i=1}^{q^a_k-1} \dot{h}^a_{i,j,k} / (q^a_k - 1)$ are the mean effective departure and arrival headways at station s_k during run r_j, and finally $\widetilde{h}^d_k \triangleq \frac{1}{n}\sum_{j=1}^{n} \dot{h}^d_{j,k}$ and $\widetilde{h}^a_k \triangleq \frac{1}{n}\sum_{j=1}^{n} \dot{h}^a_{j,k}$ are the mean effective departure and arrival headways at station s_k for the simulation campaign (which is a mean of means for all runs). Notations are summarized in Table 7.1.

Figure 7.6 shows departure headways from Los Héroes station in running direction 1. Abscissas depict events indexes, and ordinates the effective departure headways for one simulation run. Reference headways are depicted in gray and effective headways in black. One can observe that the regulation has an effect on headways. Indeed the curves of reference and effective headways are different, but their general profile remains close (there is no divergence in the effective headway curve). Now, one cannot draw conclusions

from a single run of a stochastic simulation. In what follows, we give confidence intervals for means of deviations between mean effective departure headways and mean reference headways per station, derived from a simulation campaign of several runs (here, 100).

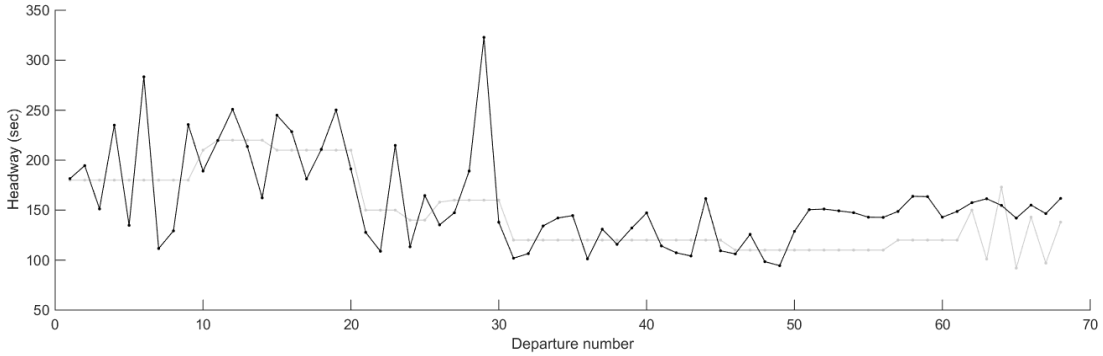


Figure 7.6: Effective and reference headways for st. Los Héroes, dir. 1 for one run

Simulation campaign

A stochastic simulation campaign can be used to estimate KPIs, with each KPI being defined as the mean value M of some quantities ζ_i measured for each simulation run r_i ($\forall i \in \{0,1,\ldots,n\}$). We call $M \triangleq \frac{1}{n}\sum_{i=1}^{n}\zeta_i$ the *sample mean* of these quantities, and σ the corresponding *estimated standard deviation*. As M is only the mean value for the performed samples, it does not necessarily reflect the real mean μ for the studied KPI. It is, thus, interesting to know how the computed value approaches μ. We compute a *confidence interval* $I \triangleq [\alpha,\beta]$ that ensures that the real mean μ belongs to interval I, with a certain confidence level c (see Section 2.3.2 to see how to compute a confidence intervals for a chosen $c \in [0,1]$).

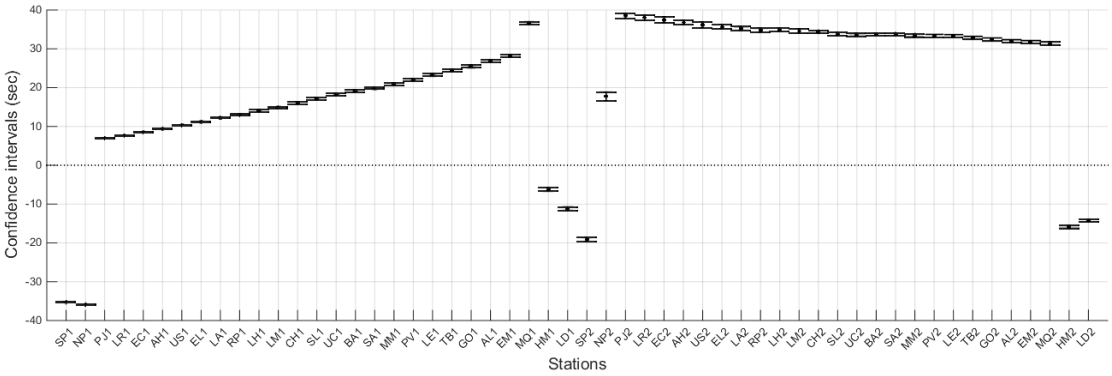


Figure 7.7: 99.9% confidence intervals for means of deviations between mean effective departure headways and mean reference headways per station

Let us now consider a KPI measuring the mean deviation w.r.t. reference departure headways for a station. The *headway deviation* for a departure event $e^d_{i,j,k}$ is defined as $\theta^d_{i,j,k} \triangleq \dot{h}^d_{i,j,k} - h^d_{i,k}$. It is the difference between the effective headway and the reference headway. The mean departure headway deviation in a run r_j at station s_k is given by $\bar{\theta}^d_{j,k} \triangleq \frac{1}{q^d_k - 1}\sum_{i=1}^{q^d_k-1}\theta^d_{i,j,k}$. Finally, the mean headway deviation at station s_k for a simulation campaign of n runs is $\bar{\theta}^d_k \triangleq \frac{1}{n}\sum_{j=1}^{n}\bar{\theta}^d_{j,k}$. The estimated standard deviation for this

KPI in a simulation campaign of n runs is $\sigma_k \triangleq \sqrt{\frac{1}{n-1}\sum_{j=1}^{n}\left(\bar{\theta}_k^d - \bar{\theta}_{j,k}^d\right)^2}$.

Figure 7.7 shows the confidence intervals computed for departure headway deviations at each station. The parameters of the simulation are $n = 100$ runs, and the intervals are computed for a confidence $c = 0.999$ (i.e., 99.9%). In this Figure, abscissae represent labelled stations, and ordinates give values of mean deviations w.r.t. reference departure headway at the considered station. For each station, the graphics contain an interval around the sample mean value computed from the simulation campaign (and symbolized by ↧). One can notice that headway deviations grow progressively from station Pajaritos dir. 1 to Manquehue dir. 1, and from Manquehue dir. 2 to Pajaritos dir. 2. This is explained by an accumulation of delays due to bottlenecks at both ends of the network. One can also notice that mean headway deviations at the ends of the line (stations SP1, NP1, HM1, LD1, SP2, NP2, HM2, and LD2) do not follow this general profile (they have smaller effective headways). This is due to the fact that, at these stations, departures are handled with more flexibility, as they are used for train insertions and turn-back maneuvers. Accumulated delays can be recovered at these stations (up to a certain limit) by considerably reducing dwell times or using fast turn back techniques. Last, one can see that the chosen disturbance level for this simulation is too high to allow recovery from delays by the selected regulation.

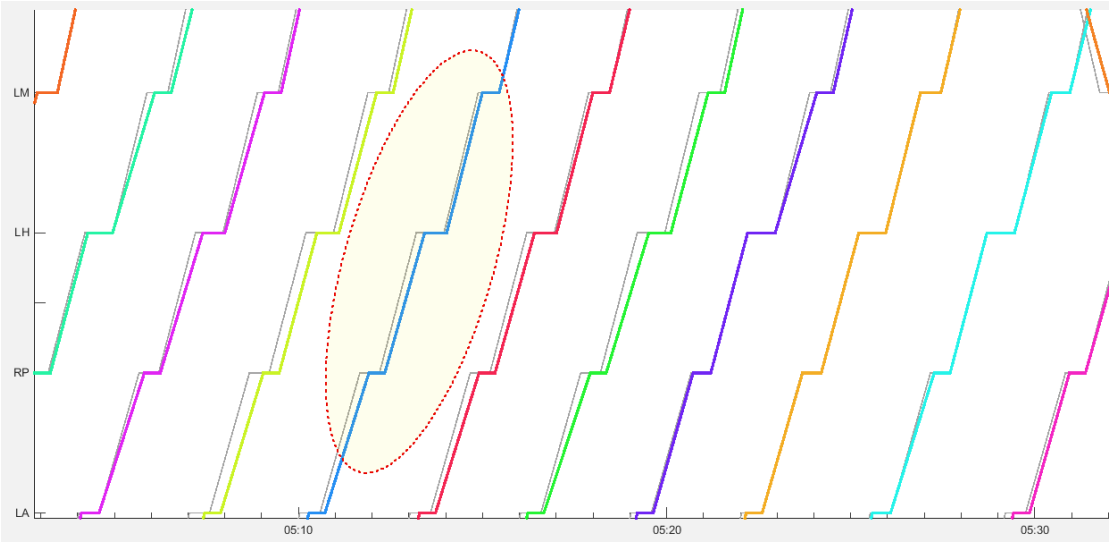


Figure 7.8: Illustration of delay recovery by a traffic management algorithm

In the case where small delays are selected, i.e., distributions are tight around nominal values of dwell and running times, the adopted traffic management technique is able to rapidly recover from deviations. This is illustrated in Figure 7.8. This Figure represents a time-space diagram in which thin gray lines are nominal trajectories of trains, and thick colored lines are the effective trajectories, i.e., the realized trajectories during a simulation run. When these two types of trajectories are superimposed, it means that the corresponding train events have occured at their nominal dates, and when shifted, it means that the train was delayed (or in advance). In the example run shown in Figure 7.8, and particularly in the circled area, one can see that the train was delayed and could get back to its nominal behavior after a few stations. This phenomenon occurs all along this simulation in which deviation values are small and shows the efficiency of the schedule

policy for traffic management when delays are small.

$e_{i,j,k}$	i^{th} event at station st_k in run r_j
$e^d_{i,j,k}$	i^{th} departure from station st_k in run r_j
$e^a_{i,j,k}$	i^{th} arrival at station st_k in run r_j
q^d_k	total number of departures from station st_k
q^a_k	total number of arrivals at station st_k
$d_{i,j}$	reference date of event $e_{i,j,k}$
$\dot{d}_{i,j,k}$	effective occurrence date of event $e_{i,j,k}$
$\delta_{i,j,k}$	timetable deviation for event $e_{i,j,k}$
$h^d_{i,k}$	reference headway for the $(i+1)^{\text{th}}$ departure from station st_k
$h^a_{i,k}$	reference headway for the $(i+1)^{\text{th}}$ arrival at station st_k
$\dot{h}^d_{i,j,k}$	effective headway for the $(i+1)^{\text{th}}$ departure from station st_k in run r_j
$\dot{h}^a_{i,j,k}$	effective headway for the $(i+1)^{\text{th}}$ arrival at station st_k in run r_j
$\bar{h}^d_k$	mean reference departure headway at station st_k
$\bar{h}^a_k$	mean reference arrival headway at station st_k
$\widetilde{h}^d_{j,k}$	mean effective departure headway at station st_k for run r_j
$\widetilde{h}^a_{j,k}$	mean effective arrival headway at station st_k for run r_j
$\widetilde{h}^d_k$	mean effective departure headway at station st_k for all runs
$\widetilde{h}^a_k$	mean effective arrival headway at station st_k for all runs
$\theta^d_{i,j,k}$	i^{th} departure headway deviation at station st_k for run r_j
$\bar{\theta}^d_{j,k}$	mean departure headway at station st_k for run r_j
$\bar{\theta}^d_k$	mean departure headway at station st_k for all runs
σ_k	estimated standard deviation at station st_k for all runs

Table 7.1: Notation summary

7.4 Discussion and improvements

The simulation campaign performed in this Chapter was done with a model of a real metro line namely, the Line 1 of Santiago's Metro, with a hold-on regulation policy that tries to stick as much as possible to a predetermined timetable. This first experimentation allowed to obtain simulation results within a reasonable time (a few seconds of simulation for 4 hours of operation of a real network, i.e., a real topology with its actual train fleet). This shows feasibility of a simulation approach to evaluate performance of regulation algorithms. Now, this simulation framework can be improved along several directions.

Distributions: First of all, distributions for delays were designed from an a priori knowledge of normal dwell and running times between two stations extracted form existing logs of Santiago's metro. Such logs are not always available for all lines To guarantee that these distributions are accurate enough, STPNs models and in particular distributions

should be built from data collected after a long enough period of observation of trains and passengers behaviors. This was the case for Santiago's Metro line 1, but not necessarily for any line under construction, or where signaling was changed recently.

A second issue regarding distributions is that the delays are modeled as Markovian noise. In this setting, every delay is sampled independently from the others. In urban train networks, latencies are correlated. For instance, if a train gets late, more passengers will enter the train, which will increase the chances of incidents, and hence of delay. Similarly, if a train is delayed due to bad weather between two stations, all trains of the network are likely to be delayed on the same part of the network. This means that sampling in our simulator should consider a context, and that distributions should be conditional distributions of the form $p(x \mid c_1, c_2, \ldots, c_k)$ where x is a delay, and $c_1, c_2, \ldots, c_k$ are variables representing the context (station, weather, day of the week, time of the day, etc.) in which delay x is sampled. This change does not require much effort to be integrated to our simulation model. However, it requires a lot of effort from designers to evaluate the impact of an environmental factor on the distributions.

Train fleets: Another issue that should be considered is the impact of fleet composition on performance of UTS. In the simulation that we have performed, we have considered regulation techniques that cannot change composition of fleets to meet their objectives. The number of running trains changes according to the period of the day, but follow planned insertions and removals of trains: It would be interesting to consider regulation techniques that can recommend to insert or remove trains to meet a desired KPI. In a similar way, we have considered uniform fleets, i.e. trains all have the same commercial speeds, the same length, capacities,etc. . This is however not the case in a real networ: trains can have different speeds, capacities, etc. One can easily integrate to distributions (and to the context as described above) the type of each train when sampling a dwell or running duration. As for all environmental factors, this difference between trains can be easily defined using conditional distributions, but with an increased design cost. However, such improvements should not penalize the efficiency of our simulation framework.

Moving block: The experiments shown in this chapter use models for a line with a fixed block policy, forbidding trains to enter an already occupied track section (block). However, many lines follow a moving block policy [55]. In our simulation framework, the STPN with blocking semantics can be replaced by a trajectory net. Changing the Petri net part of the simulation framework to adapt to this change is currently under study.

Sampling: Currently, the sampling technique for an expolynomial probability density function f with domain $[u, v]$ uses a discretization of the cumulative distribution function $F(x) = \int_0^x f(y)$. That is we obtain a set of values $x_1, \ldots x_K$, where K in the number of slices for our cumulative function and $x_i = (v - u) \cdot i/K$. Then after sampling a value η from the uniform distribution, we select the discrete value $z = x_i$ such that η lays within $[F(x_i), F(x_{i+1})]$. The sampling technique can result in a loss of accuracy if the probability density function is too roughly discretized, or in a loss of performance if the sampling technique uses a too fine discretization level and consequently sampling takes too much time.

Regulation: The regulation considered in this chapter is a simple policy that tries to stick to a predetermined timetable. The architecture of our tool uses regulation algorithms as a particular module and replacing the current regulation by another one is

quite simple. Currently, what our regulation does is: first receive an arrival date for a train, then compare it with the expected date in a timetable. Last it propagates the delays and taken decisions to the yet unexecuted part of the timetable. Changing this regulation for another one (for instance, one that tries to maintain headways between trains) within this architecture is an easy task, and other regulation techniques are currently under implementation.

Passengers flows: the last aspect that may improve accuracy of the model is to consider how passengers transfer from one line to another. Indeed, metro networks are often composed of several interconnected lines. A flow of passengers entering a line at an endpoint is likely to transfer to another line at a junction point of the network. This flow of passengers is often captured with Origin-Destination Matrices, in which entries indicate the proportion of passengers alighting at station i that leave a train at station j, or which proportion of passengers leaving at a junction station enter the next train of another line. In its current status, our model does not integrate flows nor address the number of passengers. As already mentioned, the number of passengers impacts the distribution of delays. However, integrating passengers flows to our model is likely to increase simulation time dramatically, as it requires counting (or at least quantizing) trains population, and remembering passengers alighting histories to guarantee faithful representation of passengers flows. An inspiration for this improvement of our model and of our simulation framework is the *multiphase fluid Petri nets* proposed in [30]. Another difficulty in flows representation is that Origin-Destination matrices are not known a priori. They are not available at early design stages. They have to be built once a metro network is operational, which usually requires observation of passenger habits for long periods of time.

7.5 Conclusion

In this chapter, we have detailed a framework for performance evaluation of regulation algorithms on a particular metro line. This framework consists of a high-level model of the network and of train moves, with random perturbations, in which a regulation algorithm is inserted to correct these delays. The overall systems allow for fast simulation, and hence for realization of simulation campaigns to obtain statistics on the efficiency of a regulation algorithm to meet KPI objectives.

The proposed framework allowed us to derive statistics for a case study, namely Line 1 of Santiago's Metro. A key question raised by our study is the tradeoff between abstraction (allowing efficiency of simulation) and accuracy of the statistics derived. Petri nets allow for an accurate modeling of network topologies. The more challenging part to build metro models is hence to find accurate distributions for running and dwell times. Of course, at early stages of design, one can rely on expected characteristics of the network and trains to design distributions a priori. As explained in Chapter 5, truncated expolynomial functions allow for precise modeling of distributions in which trains are more likely to be delayed than advanced. When sampling for such functions is too time consuming, these functions can be approximated, but at the cost of a loss of precision.

For an existing system, when the challenge is not design but rather to adapt regulation of train fleets and their paths to improve KPIs, one may want to work with accurate distributions, that consider elements from context: passengers, trains, but also regulation

itself. In such a situation, collected logs can help learning parameters of a distribution for dwell or running time, but it remains a challenging task to estimate the contribution of passengers or regulation decisions to the duration of a dwell time, as these parameters are usually not remembered in logs. As a future work, we plan to use our tool to compare regulation techniques, and to improve its accuracy by learning distributions.

8

Conclusion

8.1 Contribution summary

This work has led to the development of models for urban rail systems, with a variant for moving block systems and one for fixed block systems. A fully operational simulator was developed based on these models: it takes as inputs: a rail topology, a description of distributions for dwell and running time of trains (minimum, nominal and maximum times), a set of trains and their initial positions, a representation of disturbances (cumulative distribution functions), a description of a reference timetable that defines an ordering on train operations and dates. This simulator allows for efficient simulation (4 hours of operation in a few seconds on a standard PC). A real case study was modeled using original data from Santiago's metro line 1. The observed simulated behaviors were close to those observed in real logs of the line.

The second contribution of this thesis is a study of the realizability and of the robustness of reference timetables. The proposed technique relies on transient analysis, unfolding, and systems of inequalities. This method was not implemented, but it is already clear that the proposed algorithms have a high complexity. So, practical application of realizability should focus on realizability of short schedules on limited sets of events.

The last contribution of the thesis is a demonstration of the practical use of our simulation framework for the evaluation of performance of regulation, through simulation of an existing metro line and computation of statistics .

These works have led to the publications of journal and conference articles [a,b,c,d].

8.2 Perspectives and discussions

This section comments several aspects of our proposed models and approaches, analyzes their limits, and gives perspectives for future research, with a distinction between short and long term goals.

8.2.1 Short term Perspectives

Distributions. In our models, random durations, delays, etc are represented by probability density functions, and are built based on certain assumptions (e.g., delays occur with higher probability in comparison with that of advance), and from knowledge acquired by analyzing data from past operation (logs) of systems already deployed. Recalling that our simulation framework is meant to be used at early stages of projects, such a priori knowledge is not always available, and PDFs may have to be chosen arbitrarily by an experienced designer. However, the functions used in this case may not accurately reflect the stochastic behavior of the real system once deployed. In fact, this behavior will depend on many unknown parameters and will only be truly well known and understood much after the system is effectively deployed. A natural question is whether a priori distributions are realistic enough to take definitive decisions on the effectiveness of a regulation technique, or if definitive decisions shall be taken after several rounds of experiments populating a model with distributions learnt from real field data. Now, a very important point to note is that directly inferring distributions of dwell durations, for instance, from raw real field data (i.e., logged departures and arrivals over a period of time) does not always gives a faithful distribution, as the dwell times logged always reflect both regulation decisions (which are not logged) and perturbations. Another risk is to consider secondary delays as primary delays caused by incidents. This shows that even when logs are available, one needs clever learning algorithms to derive distributions of delays.

The last point of criticism for our models is that the models of STPN and trajectory Petri nets suppose that random variables that model dwell and running times (and deviations) are independent. However, this is not always the case in real situations. Indeed, there are some correlations between delays and some elements of the system: for instance, when the number of passengers increase at a given station, the chances to observe a delayed departure increase too. This delayed departure causes more passengers accumulation at the next station, and hence also impacts the next dwell duration. And so on... The models proposed in this thesis do not capture these dependencies. This could be solved by upgrading the models with an explicit representation of passengers. For this purpose, origin-destination (OD) matrices can be used. They provide information on passenger flows, and delays could depend on these matrices. However, the crux of the problem would be that faithful matrices can only be built after observing a running metro network. Generally, these data are collected through extensive surveys, but they could also be obtained from crowd mobile phone data [46]. Once a model for passengers is established, it suffices to replace the CDFs in our model by a random deviation generator based on our passenger behavior model. One must, however, always take into account the performance price to pay while upgrading this model, as a big loss of performance would make it unsuitable for Monte-Carlo simulations. Finally, note that even traffic management techniques could have some effect on deviations originating from passenger behavior. Indeed, a technique performing poorly at a station may force some passengers to choose other transportation means, i.e., reduce the number of passengers in station. Currently, our model does not address such psychological aspects of regulation.

Moving block modeling. In the trajectory Petri net (TjPN) model, safety headways are represented as fixed distances. Although practical, this choice is an oversimplification of the actual distances that must be kept between trains, and necessarily leads to

discrepancies between the simulated and real movements of trains. In reality, a headway h between two trains is generally defined as function of several parameters, two of which are the instantaneous speeds v_0 and v_1 of an upstream and a downstream train. Abstracting safety headways to fixed distances improves computation time, and significantly simplifies the adaptation of trajectories in a discrete event simulation context. In fact, as speeds of trains change with time, and as headways depend on speed, the optimal trajectory is the trajectory that maintains at any instant the highest possible speed meeting instantaneous headway constraints. In the current setting, trajectories of trains are *adapted* to the preceding trajectory in order to leave a train at a distance greater than a headway h from its predecessor. With dynamic headways, computing an adaptation for a new trajectory could be a more involved operation.

Traffic management policies. In SIMSTORS' simulation framework, the traffic management component is an algorithm that is called upon occurrence of a triggering event (e.g., the detection of a deviation from the timetable), and that updates the active timetable. In our study, we have implemented a traffic management technique that aims at recovering from delays, as early as possible, by reducing dwell times. More generally, any traffic management technique can be seen as a function that takes a timetable and computes an achievable timetable from the current state of the system, with some performance objective. A next step is the implementation of other traffic management techniques such as headway deviation minimization or rule-based proprietary techniques. This will help assessing and comparing their performances w.r.t. different criteria, and also increase the confidence in the level of abstraction in our model. Another interesting aspect is to integrate planning and optimization techniques to provide accurate timetables on a bounded duration.

Timetable robustness. The set of constraints that define the realizable behavior of a given STPN (see Capter 6 with elementary semantics is not minimal. Some of the redundancies were deliberately used to improve readability of the proposed algorithms. In particular, one does not need all event variables nor all condition variables. When it comes to implementing our proposed methods, all unnecessary variables shall be removed from the constraints associated to a pre-process. Although complexity questions were not addressed so far, the cost of the proposed realizability verification approach is expected to be high. In fact, the size of an unfolding can grow exponentially w.r.t. its depth, and checking satisfiability of a set of constraints with disjunctions can also be costly. Satisfiability of constraints is not monotonous and, hence, cannot be used to stop unfolding. However, embedding verification and unfolding can be done jointly: one can stop a branch of unfolding as soon as a schedule does not embed (without considering time) in the pre-process built along a branch of the unfolding.

Additionally, computation of realization probability for processes can be improved. We use a transient tree construction [39] that builds a symbolic but interleaved representation of some processes. This clearly comes with a very high performance cost. Evaluating probabilities of symbolic processes in a noninterleaved setting is subject for further research. It is however well known that mixing concurrency and probabilities is a difficult challenge.

8.2.2 Long term Perspectives

Multicriteria optimization. The problems addressed in this work focus on timing objectives. In mathematical optimization terms, solving a problem amounts to minimizing an objective function, such as the mean deviation for all events in a timetable or the maximum deviation for each trip (defined from a terminus to another). In reality, timing factors are not the only criteria that matter. In fact, an important criterion is *energy consumption*. Other criteria could be passenger comfort or avoidance of emergency braking... Once these criteria are mathematically formulated, it is possible to assign a weight to each criterion and to define a unique objective functions or KPIs. One would then aim to operate the system such that these KPIs are optimal. A next step would be to elaborate multicriteria traffic management techniques (for instance minimization of delays with minimization of energy consumption), and use SIMSTORS to find the right equilibrium between these objectives.

Control synthesis. The main objective of the simulation scheme presented in this thesis is evaluation and comparison of performances of traffic management policies. These regulation techniques are mainly rule-based ("if delay exceeds 10s, then reduce dwell time"). When a set of policies already exists, one only has to compare their performance and choose the most adapted regulation. However, evaluating performance of a regulation does not says how to improve it. In our models, the part of the model depicting the physical system is guided through control places, that are filled at dates prescribed by a timetable. An interesting research direction is to synthesize control laws that will replace the traditional rule-based algorithms. It is not clear whether regulation can always be expressed as timetable transformation rules, or if one needs to add ingredients to the network (control states, for instance) to obtain an optimal regulation. Some Petri net snippets that ensure safety on portions of a network, or allow to impose a given order of passage of trains were already presented in examples of Chapter 6; other more complex snippets could be developed to improve regulation.

Testing. Up to now, SIMSTORS was only tested on Santiago's metro line 1, and on some toy examples. This topology is bidirectional single line-shaped with possible turnbacks at some central crossing points (intertwined loops). Although testing was successful, this line is not the most complex existing topology. In fact, complexity in traffic management generally increases with the presence of forks and junctions or shared tracks between lines. Hence, further testing on lines and networks with different shapes, e.g., y-shaped (Paris' line 13), with shared tracks (Amsterdam's urban network), etc. could help finding the limits of our models and help consolidate them.

Realizability of schedules. In our study of timetable robustness, control actions are not taken into account. The proposed analytical approach takes a timetable and checks if it can resist to slight disturbances on events' occurrence dates. We say that a timetable is realizable up to some imprecision δ, when all events occur at their reference dates plus or minus δ, with a high enough predefined probability. It is important to note that this approach assumes that occurrence dates of events are only dictated by the CDFs attached to transition of the STPN representing the system. In reality, these dates also depend on

a traffic management policy. A major improvement for this approach would be to consider the effects of regulation in realizability questions.

Finally, traffic management in urban transportation systems should be considered in a multi modal context and not only at individual metro lines level. Dependences between passenger flows in bus systems and urban rail systems should be identified, and control policies designed to optimize the fluidity of passengers flows at the level of a whole city. Such objectives are becoming more and more attainable with the rise of big data and advanced computing, but call fro the development of new models and techniques.

Acronyms

BPN batches Petri net 50

CBTC communication-based train control 35

CDF cumulative distribution function 12, 14, 16, 48, 49, 74, 122, 134, 136

CLT central limit theorem 15–17

CPN continuous Petri net 50, 52

CPU central processing unit 123

CRP conflict resolution problem 42

DAG directed acyclic graph 25

DES discrete-event system 7, 43, 52

FB fixed-block 9, 34, 35, 40

FME Fourier–Motzkin elimination 18, 108

GBPN generalized batches Petri net 50–52

GSPN generalized stochastic Petri net 49

HPN hybrid Petri net 50, 52

JSP jobshop scheduling problem 41

KPI key performance indicator 9, 27, 31–33, 119, 120, 127, 128, 136

LLN law of large numbers 16

MB moving-block 9, 34, 35

MC Monte-Carlo 15, 16, 18

MILP mixed integer linear programming 38

OD origin-destination 134

PDF probability density function 11–14, 57, 88, 108, 134

RTS rapid transit system 37, 86, 103

SPN stochastic Petri net 48, 49

STPN stochastic time Petri net 7, 9, 49, 50, 55, 60, 70, 83, 88–90, 92, 93, 95, 96, 98, 101–103, 105–107, 109, 110, 120–123, 134–136

TdPN timed Petri net 47

TjPN trajectory Petri net 134

TPN time Petri net 47, 90, 92, 93, 108

TSD time-space diagram 82

UITP international organization for public transport 31–33

URS urban rail system 5–10, 23, 25, 31, 34, 55, 56, 58, 93, 122

Publications

Conference proceedings

[a] HÉLOUËT, L. AND KECIR, K., *Realizability of schedules by stochastic time Petri nets with blocking semantics*, In *International Conference on Applications and Theory of Petri Nets and Concurrency* (2016), Springer, pp. 155–175.

[b] ADELINE, B., D'ARIANO, A., DERSIN, P., HÉLOUËT, L. AND KECIR, K., *From reactive to predictive regulation in metros*, In *European Conference on Stochastic Optimization* (2017).

[c] ADELINE, B., DERSIN, P., FABRE, E., HÉLOUËT, L. AND KECIR, K., *An efficient evaluation scheme for KPIs in regulated urban train systems*, In *International Conference on Reliability, Safety and Security of Railway Systems* (2017), Springer, pp. 195–211.

Journal paper

[d] HÉLOUËT, L. AND KECIR, K., *Realizability of schedules by stochastic time Petri nets with blocking semantics (Extended version)*, *Science of Computer Programming 157* (2018), 71–102.

Bibliography

[1] ABDULLA, P. A., AND NYLÉN, A. Timed petri nets and bqos. In *Application and Theory of Petri Nets 2001, 22nd International Conference, ICATPN 2001, Newcastle upon Tyne, UK, June 25-29, 2001, Proceedings* (2001), J. M. Colom and M. Koutny, Eds., vol. 2075 of *Lecture Notes in Computer Science*, Springer, pp. 53–70.

[2] ADLER, G. *Die Verkettung der Streckenbelegungen und der Belegungsgrad einer Gesamtstrecke*. PhD thesis, 1968.

[3] AJMONE MARSAN, M., CONTE, G., AND BALBO, G. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems (TOCS) 2*, 2 (1984), 93–122.

[4] ALLA, H., AND DAVID, R. Continuous and hybrid Petri nets. *Journal of Circuits, Systems, and Computers 8*, 01 (1998), 159–188.

[5] ASSOCIATION OF PUBLIC TRANSPORT, I. Metro service performance indicators, a UITP information sheet, 2011.

[6] AURA, T., AND LILIUS, J. Time processes for time Petri nets. In *International Conference on Application and Theory of Petri Nets* (1997), Springer, pp. 136–155.

[7] BAUSE, F., AND KRITZINGER, P. *Stochastic Petri nets*. Verlag Vieweg, Wiesbaden, 1996.

[8] BEHRMANN, G., DAVID, A., AND LARSEN, K. A tutorial on UPPAAL 4.0, 2006.

[9] BERTHOMIEU, B., AND DIAZ, M. Modeling and verification of time dependent systems using time Petri nets. *IEEE Transactions on Software Engineering 17*, 3 (1991), 259–273.

[10] BRITANNICA, E. Rapid transit: https://www.britannica.com/technology/rapid-transit. Accessed: 2018-07-16.

[11] BÜKER, T. Methods of assessing railway infrastructure capacity. *Journal of Engineering Science & Technology* (2013), 59–71.

[12] CACCHIANI, V., CAPRARA, A., AND TOTH, P. Scheduling extra freight trains on railway networks. *Transportation Research Part B: Methodological 44*, 2 (2010), 215–231.

[13] CARNEVALI, L., GRASSI, L., AND VICARIO, E. State-density functions over DBM domains in the analysis of non-Markovian models. *IEEE Transactions on Software Engineering 35*, 2 (2009), 178–194.

[14] CHATAIN, T., AND JARD, C. Symbolic diagnosis of partially observable concurrent systems. In *FORTE'04* (2004), vol. 3235 of *LNCS*, pp. 326–342.

[15] CHATAIN, T., AND JARD, C. Time supervision of concurrent systems using symbolic unfoldings of time Petri nets. In *FORMATS'05* (2005), vol. 3829 of *LNCS*, pp. 196–210.

[16] CHATAIN, T., AND JARD, C. Complete finite prefixes of symbolic unfoldings of safe time Petri nets. In *ICATPN'06*. 2006, pp. 125–145.

[17] COXON, S., BURNS, K., AND DE BONO, A. Design strategies for mitigating passenger door holding behavior on suburban trains in Paris.

[18] D'ARIANO, A., PACCIARELLI, D., AND PRANZO, M. A branch and bound algorithm for scheduling trains in a railway network. *European Journal of Operational Research 183*, 2 (2007), 643–657.

[19] D'ARIANO, A., AND PRANZO, M. An advanced real-time train dispatching system for minimizing the propagation of delays in a dispatching area under severe disturbances. *Networks and Spatial Economics 9*, 1 (2009), 63–84.

[20] D'ARIANO, A., PRANZO, M., AND HANSEN, I. A. Conflict resolution and train speed coordination for solving real-time timetable perturbations. *IEEE Transactions on Intelligent Transportation Systems 8*, 2 (2007), 208–222.

[21] DAVID, R., AND ALLA, H. Continuous Petri nets. In *8th European Workshop on Application and Theory of Petri Nets* (1987).

[22] DEMONGODIN, I. Generalised batches Petri net: hybrid model for high speed systems with variable delays. *Discrete Event Dynamic Systems 11*, 1-2 (2001), 137–162.

[23] DEMONGODIN, I. Modeling and analysis of transportation networks using batches Petri nets with controllable batch speed. In *International Conference on Applications and Theory of Petri Nets* (2009), Springer, pp. 204–222.

[24] DEMONGODIN, I., AUDRY, N., AND PRUNET, F. Batches Petri nets. In *Systems, Man and Cybernetics, 1993.'Systems Engineering in the Service of Humans', Conference Proceedings., International Conference on* (1993), vol. 1, IEEE, pp. 607–617.

[25] ENGELFRIET, J. Branching processes of petri nets. *Acta Inf. 28*, 6 (1991), 575–591.

[26] ESPARZA, J., RÖMER, S., AND VOGLER, W. An improvement of McMillan's unfolding algorithm. *Formal Methods in System Design 20*, 3 (2002), 285–310.

[27] EZPELETA, J., COLOM, J. M., AND MARTINEZ, J. A petri net based deadlock prevention policy for flexible manufacturing systems. *IEEE transactions on robotics and automation 11*, 2 (1995), 173–184.

[28] FINKEL, A. The minimal coverability graph for Petri nets. In *International Conference on Application and Theory of Petri Nets* (1991), Springer, pp. 210–243.

[29] FRANK, M. R., SUN, L., CEBRIAN, M., YOUN, H., AND RAHWAN, I. Small cities face greater impact from automation. *Journal of The Royal Society Interface 15*, 139 (2018), 20170946.

[30] HAAR, S., AND THEISSING, S. A hybrid-dynamical model for passenger-flow in transportation systems. In *5th IFAC Conference on Analysis and Design of Hybrid Systems, ADHS 2015, Atlanta, GA, USA, October 14-16, 2015* (2015), M. Egerstedt and Y. Wardi, Eds., vol. 48 of *IFAC-PapersOnLine*, Elsevier, pp. 236–241.

[31] HAGMAN, O. Mobilizing meanings of mobility: car users' constructions of the goods and bads of car use. *Transportation research part D: Transport and environment 8*, 1 (2003), 1–9.

[32] HANSEN, I. A. State-of-the-art of railway operations research. *WIT Transactions on the Built Environment 88* (2006).

[33] HAPPEL, O. Sperrzeiten als Grundlage für die Fahrplankonstruktion. *Eisenbahntechnische Rundschau 8*, 2 (1959), 79–90.

[34] HÉLOUËT, L., AND KECIR, K. Realizability of schedules by stochastic time petri nets with blocking semantics. In *Application and Theory of Petri Nets and Concurrency - 37th International Conference, PETRI NETS 2016, Toruń, Poland, June 19-24, 2016. Proceedings* (2016), F. Kordon and D. Moldt, Eds., vol. 9698 of *Lecture Notes in Computer Science*, Springer, pp. 155–175.

[35] HÉLOUËT, L., AND KECIR, K. Realizability of schedules by stochastic time petri nets with blocking semantics. *Sci. Comput. Program. 157* (2018), 71–102.

[36] HERTEL, G. Die maximale Verkehrsleistung und die minimale Fahrplanempflindlichkeit auf Eisenbahnstrecken. *ETR. Eisenbahntechnische Rundschau 41*, 10 (1992), 665–672.

[37] HILL, R. J., AND BOND, L. J. Modelling moving-block railway signalling systems using discrete-event simulation. In *Railroad Conference, 1995., Proceedings of the 1995 IEEE/ASME Joint* (1995), IEEE, pp. 105–111.

[38] HIRAI, T. An application of temporal linear logic to timed Petri nets. In *Workshop on Applications of Petri Nets to Intelligent System Development* (1999), pp. 2–13.

[39] HORVÁTH, A., PAOLIERI, M., RIDI, L., AND VICARIO, E. Transient analysis of non-Markovian models using stochastic state classes. *Performance Evaluation 69*, 7 (2012), 315–335.

[40] KARP, R. M., AND MILLER, R. E. Parallel program schemata: A mathematical model for parallel computation. In *8th Annual Symposium on Switching and Automata Theory* (1967), pp. 55–61.

[41] KETTNER, M., SEWCYK, B., AND EICKMANN, C. Integrating microscopic and macroscopic models for railway network evaluation. *Association for European Transport* (2003).

[42] KHANSA, W., AYGALINC, P., AND DENAT, J.-P. Structural analysis of p-time Petri nets. In *CESA'96 IMACS Multiconference: computational engineering in systems applications* (1996), pp. 127–136.

[43] KNUTH, D. *The art of computer programming, Vol. 1, Fundamental Algorithms*. Addison Wesley, 1968.

[44] KOUSTOPOULOS, H., AND WANG, Z. Simulation of urban rail operations: model and calibration methodology. In *Transport Simulation, Beyond Traditional Approaches* (2009), pp. 153–169.

[45] LIME, D., AND ROUX, O. Model checking of time Petri nets using the state class timed automaton. *Discrete Event Dynamic Systems 16*, 2 (2006), 179–205.

[46] MA, J., LI, H., YUAN, F., AND BAUER, T. Deriving operational origin-destination matrices from large scale mobile phone data. *International Journal of Transportation Science and Technology 2*, 3 (2013), 183–204.

[47] MASCIS, A., AND PACCIARELLI, D. Machine scheduling via alternative graphs. Tech. Rep. DIA-46-2000, Dipartimento di Informatica e Automazione, Università degli Studi di Roma Tre, 2000.

[48] MASCIS, A., AND PACCIARELLI, D. Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research 143*, 3 (2002), 498–517.

[49] MAYR, E. W. An algorithm for the general petri net reachability problem. In *Proceedings of the 13th Annual ACM Symposium on Theory of Computing, May 11-13, 1981, Milwaukee, Wisconsin, USA* (1981), ACM, pp. 238–246.

[50] MCMILLAN, K. L. A technique of state space search based on unfolding. *Formal Methods in System Design 6*, 1 (1995), 45–65.

[51] MCMILLAN, K. L. A technique of state space search based on unfolding. *Formal Methods in System Design 6*, 1 (1995), 45–65.

[52] MERLIN, P. M. *A study of the recoverability of communication protocols*. PhD thesis, University of California, Irvine, 1974.

[53] MOLLOY, M. K. Performance analysis using stochastic Petri nets. *IEEE Transactions on Computers 31*, 9 (1982), 913–917.

[54] NASH, A., AND HUERLIMANN, D. Railroad simulation using OpenTrack. In *Computers in Railways IX* (2004), pp. 45–54.

[55] PEARSON, L. *MOVING BLOCK RAILWAY SIGNALLING*. PhD thesis, Loughborough University of Technology, 1968.

[56] PETERSON, J. L. *Petri Net Theory and the Modeling of Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.

[57] PETRI, C. A. *Kommunikation mit Automaten*. PhD thesis, Technische Hochschule Darmstadt, 1962.

[58] RADTKE, A., AND BENDFELDT, J. Handling of railway operation problems with RailSys. In *5th World Congress on Rail Research* (2001).

[59] RAMCHANDANI, C. *Performance evaluation of asynchronous concurrent systems by timed Petri nets*. PhD thesis, Massachusetts Institute of Technology, Cambridge, 1973.

[60] RECALDE, L., HADDAD, S., AND SILVA, M. Continuous Petri nets: Expressive power and decidability issues. *International Journal of Foundations of Computer Science 21*, 2 (Apr. 2010), 235–256.

[61] REISIG, W. *Petri nets: an introduction*, vol. 4. Springer Science & Business Media, 2012.

[62] SCHWANHÄUSSER, W. *Die Besessung der Pufferzeiten im Fahrplangefüge der Eisenbahn*. PhD thesis, Institut RWTH Aachen, 1974.

[63] SEMENOV, A., AND YAKOVLEV, A. Verification of asynchronous circuits using time Petri net unfolding. In *DAC* (1996), pp. 59–62.

[64] SHRIVER, A. *Theory of linear and integer programming*. Wiley, 1968.

[65] TERZI, C. L. Modeling and analysis of urban and highway traffic systems by batches Petri nets. Master's thesis, University Politehnica of Bucharest / University of Aix-Marseille, 2008.

[66] TUFFIN, B. *La simulation de Monte Carlo*. Hermes Science Publications, 2010.

[67] VAKHTEL, S. *Rechnerunterstützte analytische Ermittlung der Kapazität von Eisenbahnnetzen*. PhD thesis, Institut RWTH Aachen, 2002.

[68] WAKOB, H. *Ableitung eines generellen Wartemodells zur Ermittlung der planmäßigen Wartezeiten im Eisenbahnbetrieb unter besonderer Berücksichtigung der Aspekte Leistungsfähigkeit und Anlagenbelastung*. PhD thesis, Institut RWTH Aachen, 1985.

A

Appendix

A.1 Proof of proposition 6.1

Proposition 6.1. Let $\mathcal{N}$ be a STPN guaranteeing time progress of δ time units (between consecutive occurrences of each transition). For every date $D \in \mathbb{R}_{\geq 0}$ and condition b in an unfolding $\mathcal{U}_n$, there exists $K \geq n$ s.t. $\{b' \in \mathsf{block}(b) \mid \mathsf{dob}(b') \leq D\}$ is contained in $\mathcal{U}_K$.

Proof. Consider a pre-process PP of $\mathcal{U}_n$, which depth is more than $\lceil \frac{D}{\delta} \rceil . |T|$ events. Every event of the unfolding appended at depth i consumes conditions that were created at depth $j < i$, and at least one condition that was produced at step i of the unfolding. Hence, for every event e_n and b_n condition created at depth n, there exists a sequence $b_0 < e_1 < b_1 < \cdots < e_n < b_n$ of events and conditions of increasing depth (and also increasing dates). With the time progress assumption, we know that every consecutive pair of events representing the same transition occurs at lest at dates that differ by δ. Hence, an event created at depth n has an occurrence date of at least $\delta . \lfloor n/|T| \rfloor$. The occurrence date of an event created at depth greater than $\frac{D}{\delta} . |T|$ is hence greater than D. The number of events and conditions created at step n and appearing in the same pre-process of $\mathcal{U}_n$ is finite (as creating an event uses exclusively at least one condition of the preceding step). It is hence sufficient to unfold a net up to depth $\frac{D}{\delta} . |T|$ to obtain the (finite) set of conditions that refer to the same place as some condition b before a given date D. ■

A.2 Proof of Theorem 6.1

Proposition 6.1. Let $\mathcal{N}$ be a STPN guaranteeing time progress of δ time units. The set of time processes executable by $\mathcal{N}$ in D time units are prefixes of time processes of $\mathcal{U}_K$, with $K = \lceil \frac{D}{\delta} \rceil \cdot |T|$ containing only events with date $\leq D$.

Proof. We will show inclusion of the set of processes in the two directions. First of all, we define an ordering on symbolic processes. Let $\mathscr{E}^s = \langle E, B, \Phi \rangle$ and $\mathscr{E}^{s\prime} = \langle E', B', \Phi' \rangle$ be

two symbolic processes. We will say that $\mathscr{E}^s \sqsubseteq \mathscr{E}^{s'}$ iff there exists an event e' such that $E' = E \cup \{e'\}$, $B' = B \cup e'^\bullet$, and $\Phi = \Phi'_{|\mathrm{Var}(E,B)}$. ■

Lemma A.1. *Let $\mathscr{E}^s$ be a symbolic process of unfolding $\mathscr{U}_K$, starting from m_0, d_0, that is satisfiable and complete. Let θ be one of its solutions guaranteeing $\forall e \in E, \theta(e) \leq D$. Then, there exists a sequence $\mathscr{E}^{s,0} = \langle E_0, B_0, \Phi_0\rangle \sqsubseteq \mathscr{E}^{s,1} = \langle E_1, B_1, \Phi_1\rangle \cdots \sqsubseteq \mathscr{E}^s$ of symbolic processes of $\mathscr{U}_K$ such that $E_0 = \emptyset$, $B_0 = \{\langle \perp, p\rangle \mid p \in m_0\}$ $\Phi_0 = \{\theta(\perp) = d_0 \wedge \bigwedge_{b \in B_0} \mathsf{dob}(b) = d_0\}$ and θ is a solution for every $\mathscr{E}^{s,i}$ and $\theta(e_i) \leq \theta(e_i + 1)$.*

Proof. We can show this property by induction on the size of prefixes of $\mathscr{E}^s$. The base hypothesis is straightforward, taking the sequence with only one symbolic process $\mathscr{E}^{s,0}$ without events. Suppose that this property is satisfied for symbolic processes up to size n, and consider a satisfiable and complete symbolic process $\mathscr{E}^{s,n+1}$ of size $n+1$. Let θ_{n+1} denote a solution for this process. A growing sequence from $\mathscr{E}^{s,0}$ to $\mathscr{E}^{s,n+1}$ exists.In this sequence, the difference between $\mathscr{E}^{s,n+1}$ and $\mathscr{E}^{s,n}$ is a single event e that is maximal in $\mathscr{E}^{s,n+1}$ w.r.t. ordering on events $\preceq$, and such that $\theta(e) \geq \theta(x)$ for every event x in $\mathscr{E}^{s,n+1} \setminus \{e\}$, and $\theta(e) \geq \mathsf{dob}(b)$ for every b in $\mathscr{E}^{s,n+1} \setminus \{e\}$. Let E^n, B^n denote the set of events in $\mathscr{E}^{s,n+1} \setminus \{e\}$. Let us denote by $\Phi_{n+1|E^n,B^n}$ the restriction of Φ_{n+1} to variables attached to events and conditions E^n, B^n. One has to remove variables $\theta(e)$, $\mathsf{dod}(b)$ for every $b \in {}^\bullet e$, and $\mathsf{dob}(b)$ for every $b \in e^\bullet$ using an elimination technique such as Fourier-Motzkin. Using the properties of elimination, θ satisfies Φ_{n+1} if and only if the restriction of θ to $\mathrm{Var}(E^n, B^n)$ satisfies $\Phi_{n+1|E^n,B^n}$. However, the restriction of θ is exactly θ_n, and as $\theta(e)$, $\mathsf{dod}(b)$ for $b \in e^\bullet$, and $\mathsf{dod}(b)$ for $b \in {}^\bullet e$ are all greater than variables in $\mathrm{Var}(E^n, B^n)$, the elimination of variables is simply a projection on atoms that do not contain variables related to e, and $\Phi_{n+1|E^n,B^n} = \Phi_n$. ■

Lemma A.2. *Given a symbolic process $\mathscr{E}^s$ of $\mathscr{U}_K$, one of its solutions θ, and an ordering $\mathscr{E}^{s,0} = \langle E_0, B_0, \Phi_0\rangle \sqsubseteq \mathscr{E}^{s,1} = \langle E_1, B_1, \Phi_1\rangle \cdots \sqsubseteq \mathscr{E}^s$ as above, then the word $u_{\mathscr{E}^s,\theta} = \langle t_1, \theta(e_1)\rangle \ldots \langle t_{|E|}, \theta(e_{|E|})\rangle$ is a timed word of $\mathscr{L}(\mathscr{N})$.*

Proof. Again we can prove this lemma by induction. The base case is obvious, as the empty word ϵ is a timed word of $\mathscr{L}(\mathscr{N})$. Let us suppose that the property is satisfied up to n, that is for every process $\mathscr{E}_n$ of size n and solution θ_n meeting all constraints of $\mathscr{E}_n$, there exists an increasing sequence of prefixes of $\mathscr{E}_n$ such that the word associated with this sequence is a timed word of $\mathscr{L}(\mathscr{N})$.

Let us now consider a time process $\mathscr{E}_{n+1}$ with $n+1$ events and one of its solutions θ_{n+1}. As in Lemma A.1, one can find an event e_{n+1} and a process $\mathscr{E}_n$ such that $\mathscr{E}_n$ and $\mathscr{E}_{n+1}$ only differ by addition of this single event. There exists a timed word $u_n = \langle e_1, \theta_{n+1}(e_1)\rangle \ldots \langle e_n, \theta_{n+1}(e_n)\rangle \in \mathscr{L}(\mathscr{N})$ corresponding to $\mathscr{E}_n$. This word may lead the net to any configurations in a set $Conf_n$ with identical markings, but distinct times to fire attached to transitions. However, as we know that θ_{n+1} meets all constraints of $\mathscr{E}_{n+1}$, there exists a configuration in $Conf_n$ whose times to fire allow firing of e_{n+1} at date $\theta(e_{n+1})$, and $u_{n+1} = u_n.\langle e_{n+1}, \theta(e_{n+1})\rangle \in \mathscr{L}(\mathscr{N})$. ■

Note that assuming time progress, the dates attached to an event of a process of $\mathscr{U}_K$ that occur at a date smaller than D cannot be further constrained by addition of constraints coming from events that are not in $\mathscr{U}_K$. The two lemmas above hence allow to conclude that for a given symbolic process $\mathscr{E}^s$ of unfolding $\mathscr{U}_K$, in which one considers events that occur before date D, and for each solution of $\mathscr{E}^s$, we have $\mathscr{E}^s_\theta = TP_{u_{\mathscr{E}^s,\theta}}$ for some word

$u_{\mathcal{E}^s,\theta} \in \mathscr{L}(\mathcal{N})$. Hence, the set of time processes of $\mathcal{U}_K$ whose events occur before D is contained in the set of time processes $TP(\mathscr{L}_{\leq D}(\mathcal{N}))$. All time processes of some pre-process of $\mathcal{U}_K$ (and hence all time processes of unfolding $\mathcal{U}_K$) can be built from a timed word that is executable by $\mathcal{N}$ in less than D time units, and are hence time processes of $\mathcal{N}$.

We now have to prove the converse direction, i.e., every time process associated with a word $u \in \mathscr{L}_{\leq D}(\mathcal{N})$ is a time process of $\mathcal{U}_K$.

Lemma A.3. *Let $u \in \mathscr{L}_{\leq D}(\mathcal{N})$. Then, $TP(u)$ is a time process of $\mathcal{U}_K$.*

Proof. We proceed by induction on the size of words. First, for the empty words, the time process with only initial conditions is clearly a time process of $\mathcal{U}_K$. Let us now assume that for every $u_n = \langle e_1,\theta(e_1)\rangle \dots \langle e_n,\theta(e_n)\rangle \in \mathscr{L}_{\leq D}(\mathcal{N})$ of length n, $TP(u_n)$ is a time process of $\mathcal{U}_K$. Let us consider a word $u_{n+1} = \langle e_1,\theta(e_1)\rangle \dots \langle e_n,\theta(e_n)\rangle.\langle e_{n+1},\theta(e_{n+1})\rangle \in \mathscr{L}_{\leq D}(\mathcal{N})$. One can build a time process $\mathcal{E}_u$ for $u = \langle e_1,\theta(e_1)\rangle \dots \langle e_n,\theta(e_n)\rangle$. Clearly, as $u_{n+1} \in \mathscr{L}_{\leq D}(\mathcal{N})$, word u leads from marking m_0 to a marking that enables e_{n+1}. Let $e_{p_1},\dots,e_{p_k}$ denote the k events that produce the tokens that are consumed by e_{n+1}. If event e_{n+1} is a firing of some transition t that occurs exactly when its time to fire has expired, $\theta(e_{n+1})$ meets the constraint $\text{eft}(t) + \max\{\theta(e_{p_i})\} \leq \theta(e_{n+1}) \leq \text{lft}(t) + \max\{\theta(e_{p_i})\}$. In any case, we have $eft(t) + \max\{\theta(e_{p_i})\} \leq \theta(e_{n+1})$ (which is the only constraint w.r.t predecessors imposed by constraint in the unfolding. Similarly, let $e_{b_1},\dots e_{b_q}$ denote the last events of u that free places in which t outputs some tokens (and hence may have blocked the execution of t before $\theta(e_{n+1})$). We have $\theta(e_{n+1})$ meets the constraint $\max(\{\theta(e_{b_i})\}) \leq \theta(e_{n+1})$. Hence, any event that had to occur before $\theta(e_{n+1})$ (due to urgency, causality, or blockings) also appears in $\mathcal{E}_u$. Hence, θ witnesses satisfiability of a set of constraints over occurrence dates of events $e_1,\dots,e_n$, and one can safely append $e_{n+1} = (B,t)$ to maximal places of $\mathcal{E}_u$, and obtain a symbolic prefix $\mathcal{E}_{u_{n+1}}$ (satisfiable, conflict free and complete). It now remains to show that $\mathcal{E}_{u_{n+1}}$ is a symbolic process of $\mathcal{U}_K$. As $\theta(e_{n+1}) \leq D$, e_{n+1} appears in the unfolding of $\mathcal{N}$ at depth at most $\frac{D}{\delta}$, which is lower than $K = \lceil \frac{D}{\delta} \rceil \cdot |T|$. Hence, $\mathcal{E}_{u_{n+1}}$ is an causally closed set of events that also contains all mandatory urgent transition firings and place unblockings whose set of constraints is satisfiable, and contained in $\mathcal{U}_K$, i.e., it is a symbolic process of $\mathcal{U}_K$. ■

A.3 Stochastic state class tree

In this part of the appendix, we detail how to build a stochastic state class tree for a particular process of a stochastic Petri net with blocking semantics.

Definition A.1 (transient stochastic state class tree)**.** A *transient stochastic state class tree* for an STPN $\mathcal{N}$ (that we shall call tree, for short) is a directed acyclic graph $S = \langle V, \circ\!\!\rightarrow \cup \bullet\!\!\rightarrow \rangle$ where vertices in V are classes, edges in $\circ\!\!\rightarrow$ represent firing transitions after (symbolically) elapsing time, and edges in $\bullet\!\!\rightarrow$ represent firings of urgent transitions. Every vertex in the tree has only one predecessor except for the root of the tree, denoted v_0, that has no predecessor. Edges carry probabilistic informations on transitions firings and the sum of probabilities of all edges leaving the same vertex is equal to 1.

The construction of a tree starts from the initial class Σ_0 (with marking m_0, a domain C_0 for the TTFs of transitions enabled in m_0 and all other components defined accordingly,

see appendix E) and inductively computes edges and reachable classes. Edges $\Sigma \xrightarrow{t,\mu} \Sigma'$ from a class Σ to a successor class Σ' are labeled by a transition name t and by the probability μ to fire t from Σ, and are of two forms:

Firing after elapsing time: A move $\Sigma \overset{t_i,\mu_i}{\hookrightarrow} \Sigma'$ from $\Sigma = \langle m, C, D, \text{BLK}, \text{URG}\rangle$ to $\Sigma' = \langle m', C', D', \text{BLK}', \text{URG}'\rangle$, achievable with probability μ_i, consists in firing transition t_i after symbolically elapsing its TTF. Such a move is only allowed if $\text{URG} = \varnothing$ and the TTF τ_i of t_i is less than or equal to TTFs of all other transitions that could fire from Σ. The time domain C^i from which t_i can fire is hence $C^i = C \cap \bigcap_{x_j \in X_M} \{x_i \leq x_j\}$, and the probability of firing t_i from Σ is $\mu_i = \int_{C^i} D$. We have $m' = m - {}^\bullet t_i + t_i{}^\bullet$. The new domain C' and distribution D' are computed as for STPNs with non-blocking semantics: it is obtained by advancing time, removing variables of disabled transitions and adding those of newly enabled transitions [39], and then removing variables of transitions whose domain is the singleton $\{0\}$. The domain of a single variable x_i in a domain C over several variables can be obtained by eliminating all variables but c_i from C. As soon as a variable has domain $\{0\}$, the time to fire of the associated transition is necessarily zero, and the transition has to fire. It is then stored in the set of urgent transitions if it is not blocked, and in the set of blocked transitions otherwise. The set BLK' is obtained by removing from BLK transitions that were disabled by firing of t_i and transitions that are not blocked anymore in m' thanks to the places freed by firing of t_i (they become urgent), and adding transitions which are enabled in m' with a firing domain in C' that is $\{0\}$. Finally, the set URG' contains all enabled transitions that became urgent when firing t_i, i.e., transitions with firing domain $\{0\}$ among enabled transitions, and formerly blocked transitions unblocked by t_i.

Firing urgent transitions: In STPNs semantics, when more than one transition is firable from a configuration, their weights are used to compute the probability of firing each transition. This case can occur because of blocking semantics: an STPN can keep several transitions blocked, and firing a transition can also unlock several of them at the same time (all unblocked transitions become urgent). When a class Σ has urgent transitions ($\text{URG} \neq \varnothing$), only moves of the form $\Sigma \overset{t_i,\mu_i}{\bullet\!\!\rightarrow} \Sigma'$ are allowed. They consist in firing a transition t_i among urgent transitions in URG with probability $\mu_i = \mathcal{W}(t_k) \big/ \sum_{t_j \in \text{URG}} \mathcal{W}(t_j)$. Components m', C' and D' of the successor class are computed as for timed moves, with the only difference that no time elapses before the firing of t_i. Set BLK' is obtained by removing from BLK transitions that are unlocked or disabled by the firing of t_i and adding those that become blocked in m', and URG' contains transitions from URG that were not disabled by firing of t_i and transitions from BLK that were unblocked when firing t_i.

Computing the probability of a process $\mathscr{E}^s$:

Transient trees are a priori infinite, but very often, one can work with a bounded horizon. This is our case when evaluating the probability of realization in $\mathscr{U}_K$. Let $\Psi_i = \{\psi_{i,0}, \psi_{i,1}, \ldots, \psi_{i,n-1}\}$ denote all possible embeddings of a schedule S into a symbolic process $\mathscr{E}^s_i$ of $\mathscr{N}$. We denote by $\mathbb{P}(\Phi_{\psi_{i,j}, d\pm\alpha})$ the probability that $\mathscr{N}$ executes a time process $\mathscr{E}^s_i$ and realizes S within a precision of $\pm\alpha$ when $\psi_{i,j}$ is the embedding of S in $\mathscr{E}^s_i$. We adapt the tree construction to consider only $\mathscr{E}^s_i$ and embedding $\psi_{i,j}$, and compute $\mathbb{P}(\mathscr{E}^s_i \wedge \Phi_{\psi_{i,j}, d\pm\alpha})$. We build a tree whose vertices of the form $\langle \Sigma, S\rangle$ memorize a class and a suffix of $\mathscr{E}^s_i$ not yet executed. We start from vertex $\langle \Sigma_0, \mathscr{E}^s_i\rangle$. We create an edge $\langle \Sigma, S\rangle \xrightarrow{t_k,\mu_k} (\Sigma', S')$ representing firing of a transition t_k if there is a minimal event e in the remaining suffix of $\mathscr{E}^s_i$, and $\text{tr}(e) = t_k$. S' is the new suffix obtained by removing e from S. Σ' is the successor class obtained after firing this transition from Σ. Edges are built as be-

fore in the tree, but with an additional constraint: edges with label t_k, μ_k and components m, C, D, BLK, URG of classes are built with the additional requirement that when creating an edge from an event e that is in the image of $\psi_{i,j}$, the firing time domain is restricted to impose that e occurs in the time interval $I_k = [\max(0, d(\psi_{i,j}^{-1}(e)) - \alpha), d(\psi_{i,j}^{-1}(e)) + \alpha]$. In this case, the probability of firing $t_k = \mathrm{tr}(e)$ becomes $\mu_k = \int_{C^k \cap C'^k} D$, where C'^k is the part of C in which $\tau_k - \tau_{\mathrm{age}}$ (the firing date for e) belongs to I_k. We stop developing a branch of the tree at vertices whose suffix does not contain events that are images of nodes in S via $\psi_{i,j}$. The construction ends with a tree $S_{i,j,\alpha}$.

Transient tree $S_{i,j,\alpha}$ measures the probability of solutions and of occurrence of a particular process. The probability $\mathbb{P}(\mathscr{E}_i^s \wedge \Phi_{\psi_{i,j},\alpha})$ is computed as $\mathbb{P}(\mathscr{E}_i^s \wedge \Phi_{\psi_{i,j},\alpha}) = \sum_{\rho \in \mathrm{PATH}(S_{i,j,\alpha})} \mathbb{P}(\rho)$ where $\mathrm{PATH}(S_{i,j,\alpha})$ is the set of paths from $\langle \Sigma_0, \mathscr{E}_i^s \rangle$ to a leaf of $S_{i,j,\alpha}$, and the probability $\mathbb{P}(\rho)$ of a path $\rho = \Sigma_0 \xrightarrow{t_i,\mu_i} \dots \xrightarrow{t_{l-1},\mu_{l-1}} \Sigma_l$ that start at Σ_0 and end on a leaf Σ_l is the product $\Pi_{i \in \{0,\dots,l-1\}} \mu_i$. As soon as $\mathbb{P}(\mathscr{E}_i^s \wedge \Phi_{\psi_{i,j},\alpha}) > 0$, S has a non-null probability to be realized (with a tolerance of $\pm\alpha$). Noticing that different embeddings yield disjoint paths in their respective transient stochastic state class trees, the probability for a schedule to be realized by a process is hence $\mathbb{P}(\mathscr{E}_i^s \models S) = \sum_{\psi_{i,j} \in \Psi_i} \mathbb{P}(\mathscr{E}_i^s \wedge \Phi_{\psi_{i,j},\alpha})$.

Finally, denoting by $\mathscr{E}(PP, S)$ the symbolic processes of a pre-process PP that embed S, the probability $\mathbb{P}(\mathscr{N} \models S)$ is greater than $\max\{\mathbb{P}(\mathscr{E}_i^s \wedge \Phi_{\psi_{i,j},\alpha}) \mid PP \in \mathrm{PP}(U_K) \wedge \mathscr{E}_i^s \in \mathscr{E}(PP, S)\}$. It is difficult to obtain more than a lower bound for realization, as symbolic processes of $\mathscr{E}(PP, S)$ might have overlapping executions.

A.4 Derivation of components of successor class

We hereafter provide details on how to compute components C' and D' of a class Σ' obtained from a class Σ through a transition $\overset{t_i,\mu_i}{\circ\!\!\rightarrow}$. Derivation of components m', BLK$'$ and URG$'$ has already been covered and need not further explanations.

We shall use the following notations:

- given a time domain C delimiting the possible values of a set of variables $X_M = \{x_0, x_1, \dots, x_{N-1}\}$, we will denote by $C{\downarrow_{x_i}}$ the projection of C that eliminates variable x_i from X_M. The elimination is done via the Fourier-Motzkin method detailed in Chapter 2.
- given a vector $\underline{x} = \langle x_0, x_1, \dots, x_{N-1} \rangle$, we denote by $\underline{x} \setminus x_i$ the vector $\underline{x}$ from which variable x_i is removed, with $i \in \{0, 1, \dots, N-1\}$;
- the addition of a scalar x_0 to each element of a vector $\underline{x} = \langle x_1, x_2, \dots, x_n \rangle$ is simply written $\underline{x} + x_0$.

Probability of firing: A transition t_i can fire from class Σ iff t_i is enabled by m and no postset place of t_i is occupied; that is $\forall p \in t^\bullet, m(p) = 0$. We also need its TTF x_i to be less than or equal to TTFs of all variables in X_M; transition t_i will then fire from Σ with probability μ^i, with:

$$\mu^i = \int_{C^i} D$$

C^i is the time domain from which t_i shall fire with all its TTF less than or equal to every

variable in X_M.

$$C^i \triangleq C \cap \bigcap_{x_j \in X_M} \{x_i \leq x_j\}$$

Precedence condition: The assumption that t_i fires before any other transition adds conditions on the time vector and thus leads to a new random variable X_M^a distributed over C^i according to the following conditional PDF:

$$D^a = D \Big/ \mu^i$$

Time elapsing and elimination: According to the semantics of STPNs, when t_i fires, TTFs of activated transitions are decreased by the value of the TTF of t_i, namely τ^i. This yields a new random variable $X_M^b = X_M^a - x_i$ distributed over the domain $C^b = C^i \downarrow_{x_i}$ in which the variable attached to the fired transition t_i is eliminated. The PDF of the new multivariate random variable X_M^b is then:

$$D^b = \int_{\text{MIN}^i}^{\text{MAX}^i} D^a$$

where MIN^i and MAX^i denote the bounds of the support of variable τ^i.

Disabling: If the firing of t_i disables a transition t_j, variable x_j has to be eliminated from the time vector, yielding a new vector $X_M^c = X_M^b \setminus x_j$ distributed over $C^c = C^b \downarrow_{x_j}$ with PDF:

$$D^c = \int_{\text{MIN}^j}^{\text{MAX}^j} D^b$$

The same procedure is repeated for every disabled transition by the firing of t_i. Let X_M^{c*}, C^{c*} and D^{c*} then respectively denote the resulting time vector, domain and PDF.

Newly enabling: If the firing of t_i enables a transition t_k, with PDF f_{t_k} over $[\text{eft}(t_k), \text{lft}(t_k)]$, then the new time vector, that we denote by $X_{M'}^d$, shall include an additional component x_k and shall be distributed over $C^d = C^{c*} \times [\text{eft}(t_k), \text{lft}(t_k)]$ according to the PDF:

$$D^d = D^{c*} \times f_{t_k}(x_k)$$

The same procedure is similarly repeated for every newly enabled transition to finally obtain the PDF of the successor class Σ'.

Titre : Évaluation de performances pour les techniques de régulation du trafic ferroviaire urbain

Mots clés : systèmes ferroviaires, régulation du trafic, réseaux de Petri, évaluation de performance, simulation stochastique, tables horaires

Résumé : Le trafic ferroviaire urbain est quotidiennement sujet à des perturbations qui le dévient de son comportement nominal. Afin de minimiser l'impact de ces perturbations, les opérateurs ferroviaires usent de diverses techniques. Nonobstant leur efficacité, les performances de ces techniques ne sont généralement pas bien étudiées ni sont-elles optimales, car élaborées empiriquement. C'est dans ce cadre-ci que vient cet ouvrage fournir des solutions qui permettent d'évaluer ces techniques de régulation et d'en comparer les performances dans des contextes variés.
L'approche proposée se base sur des variantes de réseaux de Petri comme modèles et sur la méthode de Monte-Carlo pour en simuler l'exécution.
Cette combinaison a donné naissance à SIMSTORS, un outil de simulation pour les systèmes ferroviaires urbains, et plus généralement, pour les systèmes stochastiques régulés.
Additionnellement, nous nous intéressons dans cette thèse à la problématique de la réalisabilité des tables horaires qui pilotent le trafic ferroviaire. Ces tables décrivent le comportement temporel désiré des systèmes pour lesquels elles sont conçues. Or, la construction de ces tables ne garantit pas toujours sa réalisabilité, notamment dans un contexte stochastique. Ainsi, nous proposons ici une méthode permettant de vérifier si une table horaire est bien réalisable avec une probabilité strictement positive.

Title: Performance evaluation of urban rail traffic management techniques

Keywords: rail systems, traffic management, Petri nets, performance evaluation, stochastic simulation, timetables

Abstract: Urban rail traffic is subject to numerous disrupting events that drift it from its nominal behavior. In order to minimize the impact of these disturbances, rail operators rely on a set of techniques. Despite their efficiency, performances of these techniques are rarely well studied, nor are they of proven optimality; a direct consequence of them being empirically built. It is in this particular context that comes our work to provide solutions that allow for the evaluation of such techniques and for the comparison of their relative performances in various scenarios. The proposed approach is based on variants of Petri nets as models, and on the Monte-Carlo method for the simulation of their execution.
This combination has led to the development of SIMSTORS, a tool for the simulation of urban rail systems, and more generally, stochastic systems under dynamic rescheduling. Additionally, this thesis addresses the question of timetable realizability; that is whether or not a given timetable is indeed realizable by a system for which it was built. Indeed, a timetable is meant to drive the behavior of a system but there is no guarantee as to its realizability. We therefore propose a method for the verification of the realizability of timetables with a strictly positive probability.