Debugging Embedded Multimedia Application Execution Traces through Periodic Pattern Mining

Patricia López Cueva

CIFRE Thesis supervised by:
Alexandre Termier, Jean François Méhaut and Miguel Santana

8th July, 2013

Context

STMicroelectronics

Leading semiconductor manufacturer.

Telecommunications and Multimedia

- Highly integrated devices.
- Competitive market: Hardware + Software.
- Software development: difficult and slow.
- Time-to-market.
- Debugging phase: long and costly.

Debugging Techniques

Software Debugging

Functional Debugging

Interactive debuggers $\Rightarrow$ High Intrusiveness

Performance Debugging

Profilers $\Rightarrow$ Not enough detail

More Parallelism

- More bugs: interaction between components of the system.
- Interactive debuggers or profilers not suitable to diagnose these bugs.

Tracing: A multipurpose solution

- Recording the execution of the application for a postmortem analysis.
- *Problem:* Their size is becoming unmanageable for a manual analysis.
- *Need:* Automatic analysis tools.

Pattern Mining: A possible solution

Data Mining

Extract knowledge from huge volumes of data.

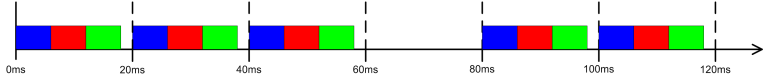
Frequent Pattern Mining

Discover regularities in the data that are called patterns.

Fine-grained analysis of the application behavior.

Characteristic of multimedia application

Periodic behavior (frame decoding).



Proposal

A new approach to debug multimedia application execution traces through periodic pattern mining.

Outline

1. Introduction
   - Context and Motivation
2. Formal Framework
   - Frequent Periodic Patterns
   - Core Periodic Concepts
   - PerMiner Algorithm
   - Scalability Experiments
3. Its Application
   - Usage Guidelines
   - Use Cases
4. Conclusions
   - Conclusions
   - Future Work

Outline

1. Introduction
   - Context and Motivation
2. Formal Framework
   - Frequent Periodic Patterns
   - Core Periodic Concepts
   - PerMiner Algorithm
   - Scalability Experiments
3. Its Application
   - Usage Guidelines
   - Use Cases
4. Conclusions
   - Conclusions
   - Future Work

What is a periodic pattern?

Periodic Pattern Group of events that appear regularly in the trace.

Pattern Mining Set of items that appear periodically in the transactional DB.

Execution Trace ($s.\mu s$)

Decode of a frame

```
68.770630 getFrame
68.770697 displayFrame
68.770741 int16
68.770768 swint16
---------------------
68.770869 displayFrame
68.770913 getFrame
68.770959 write16
68.770982 cpu_clock
---------------------
68.771032 getFrame
68.771099 displayFrame
68.771150 read16
68.771235 fork
68.771324 get_pid
---------------------
68.771346 getFrame
68.771372 displayFrame
68.771402 printk
68.771456 sem_up
68.771487 sem_down
---------------------
68.771540 getFrame
68.771586 displayFrame
```

What is a periodic pattern?

Periodic Pattern $\boxed{\text{Group of events}}$ that appear $\boxed{\text{regularly}}$ in the $\boxed{\text{trace.}}$

Pattern Mining $\boxed{\text{Set of items}}$ that appear $\boxed{\text{periodically}}$ in the $\boxed{\text{transactional DB.}}$

Execution Trace ($s.\mu s$)

68.770630 getFrame
68.770697 displayFrame
68.770741 int16
68.770768 swint16
68.770869 displayFrame
68.770913 getFrame
68.770959 write16
68.770982 cpu_clock
68.771032 getFrame
68.771099 displayFrame
68.771150 read16
68.771235 fork
68.771324 get_pid
68.771346 getFrame
68.771372 displayFrame
68.771402 printk
68.771456 sem_up
68.771487 sem_down
68.771540 getFrame
68.771586 displayFrame

0.1 ms

What is a periodic pattern?

Periodic Pattern Group of events that appear regularly in the trace.

Pattern Mining Set of items that appear periodically in the transactional DB.

Execution Trace ($s.\mu s$)

0.1 ms

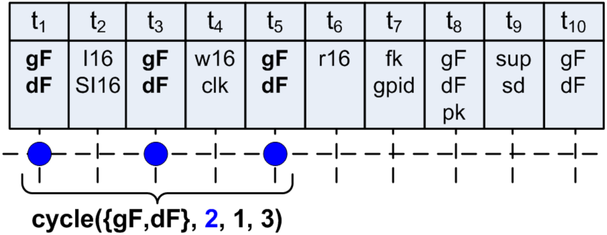
68.770630	getFrame
68.770697	displayFrame
68.770741	int16
68.770768	swint16
68.770869	displayFrame
68.770913	getFrame
68.770959	write16
68.770982	cpu_clock
68.771032	getFrame
68.771099	displayFrame
68.771150	read16
68.771235	fork
68.771324	get_pid
68.771346	getFrame
68.771372	displayFrame
68.771402	printk
68.771456	sem_up
68.771487	sem_down
68.771540	getFrame
68.771586	displayFrame

Preprocessing

Transactional Database

t_1	getFrame, displayFrame
t_2	int16, swint16
t_3	displayFrame, getFrame
t_4	write16, cpu_clock
t_5	getFrame, displayFrame
t_6	read16
t_7	fork, get_pid
t_8	getFrame, displayFrame, printk
t_9	sem_up, sem_down
t_{10}	getFrame, displayFrame

Cycle(*itemset*, *period*, *offset*, *length*)



Itemset Set of events.

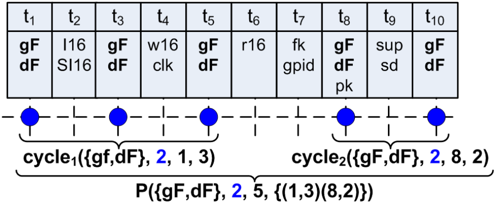
Period Distance between two consecutive transactions of the cycle.

Offset Transaction identifier of first transaction forming part of the cycle.

Length Number of transactions forming part of the cycle.

Legend : *gF* getFrame, *dF* displayFrame, *I*16 int16, *SI*16 swint16, *w*16 write16, *clk* cpu_clock, *r*16 read16, *fk* fork, *gpid* get_pid, *pk* printk, *sup* sem_up, *sd* sem_down.

Periodic Pattern $P(itemset, period, support, cycles)$



Periodic Pattern [Ma & Hellerstein, 2001]

A group of cycles forms a periodic pattern if:

1. Same period for all cycles.
2. All cycles are consecutive.
3. Cycles do not overlap.

Support

Sum of all *cycles* lengths:
$cycles = \{(o_1, l_1), ..., (o_k, l_k)\}$

$$support = \sum_{i=1}^{k} l_i$$

Frequent Periodic Pattern

Given a minimum support threshold (min_sup), a pattern is frequent if

$$support \geq min\_sup$$

How many Frequent Periodic Patterns?

High redundancy!

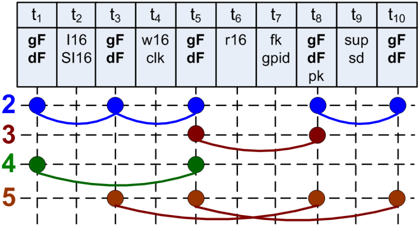
1. All combinations of a large itemset.
2. All combinations of frequent periods.

Frequent Periodic Patterns
$P_1(\{gF\}, 2, 5, \{(1,3)(8,2)\})$
$P_2(\{dF\}, 2, 5, \{(1,3)(8,2)\})$
$P_3(\{gF, dF\}, 2, 5, \{(1,3)(8,2)\})$
...
$P_6(\{gF, dF\}, 3, 2, \{(5,2)\})$
...
$P_9(\{gF, dF\}, 4, 2, \{(1,2)\})$
...
$P_{12}(\{gF, dF\}, 5, 2, \{(3,2)\})$
...
$P_{15}(\{gF, dF\}, 5, 2, \{(5,2)\})$

1

2

Triadic Approach [Lehmann et al., 1995]



Triadic Context

3 sets
+
1 ternary relation

Itemsets	Periods	2										3									
	Transactions	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
gF		X		X		X			X		X					X			X		
dF		X		X		X			X		X					X			X		
...																					

Itemsets	Periods	4										5									
	Transactions	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
gF		X				X								X		X			X		X
dF		X				X								X		X			X		X
...																					

Set of periodic concepts

	Periods	2										3									
Itemsets	Transactions	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
gF		X		X		X			X		X					X			X		
dF		X		X		X			X		X					X			X		
...																					
	Periods	4										5									
Itemsets	Transactions	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
gF		X				X								X		X			X		X
dF		X				X								X		X			X		X
...																					

Triples
$(\{gF, dF\}, \{2\}, \{t_1, t_3, t_5\})$

Set of periodic concepts

Itemsets	Periods	2										3									
	Transactions	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
gF		X		X		X			X		X					X			X		
dF		X		X		X			X		X					X			X		
...																					

Itemsets	Periods	4										5									
	Transactions	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
gF		X				X								X		X			X		X
dF		X				X								X		X			X		X
...																					

Periodic Concepts
$T_1(\{gF, dF\}, \{2\}, \{t_1, t_3, t_5, t_8, t_{10}\})$

Set of periodic concepts

	Periods	2										3									
Itemsets	Transactions	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
gF		X		X		X			X		X					X			X		
dF		X		X		X			X		X					X			X		
...																					
	Periods	4										5									
Itemsets	Transactions	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
gF		X				X								X		X			X		X
dF		X				X								X		X			X		X
...																					

Periodic Concepts
$T_1(\{gF, dF\}, \{2\}, \{t_1, t_3, t_5, t_8, t_{10}\})$
$T_2(\{gF, dF\}, \{2, 4\}, \{t_1, t_5\})$

Set of periodic concepts

Itemsets	Periods	2										3									
	Transactions	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
gF		X		X		X			X		X					X			X		
dF		X		X		X			X		X					X			X		
...																					

Itemsets	Periods	4										5									
	Transactions	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
gF		X				X								X		X			X		X
dF		X				X								X		X			X		X
...																					

Periodic Concepts
$T_1(\{gF, dF\}, \{2\}, \{t_1, t_3, t_5, t_8, t_{10}\})$
$T_2(\{gF, dF\}, \{2, 4\}, \{t_1, t_5\})$
$T_3(\{gF, dF\}, \{2, 5\}, \{t_3, t_5, t_8, t_{10}\})$

Set of periodic concepts

	Periods	2										3									
Itemsets	Transactions	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
gF		X		X		X			X		X					X			X		
dF		X		X		X			X		X					X			X		
...																					

	Periods	4										5									
Itemsets	Transactions	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
gF		X				X								X		X			X		X
dF		X				X								X		X			X		X
...																					

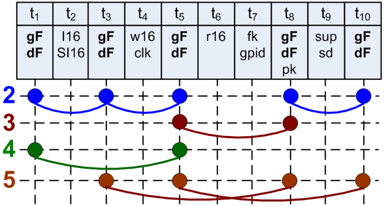
Periodic Concepts
$T_1(\{gF, dF\}, \{2\}, \{t_1, t_3, t_5, t_8, t_{10}\})$
$T_2(\{gF, dF\}, \{2, 4\}, \{t_1, t_5\})$
$T_3(\{gF, dF\}, \{2, 5\}, \{t_3, t_5, t_8, t_{10}\})$
$T_4(\{gF, dF\}, \{2, 3, 5\}, \{t_5, t_8\})$

Core Periodic Concepts (CPC)

Core Periodic Concept

A periodic concept (I, P, T) is a **core periodic concept** if there does not exist any other periodic concept (I', P', T') such that $I = I'$, $P' \subset P$ and $T' \supset T$.

Periodic Concepts
$T_1(\{gF, dF\}, \{2\}, \{t_1, t_3, t_5, t_8, t_{10}\})$
$T_2(\{gF, dF\}, \{2, 4\}, \{t_1, t_5\})$
$T_3(\{gF, dF\}, \{2, 5\}, \{t_3, t_5, t_8, t_{10}\})$
$T_4(\{gF, dF\}, \{2, 3, 5\}, \{t_5, t_8\})$



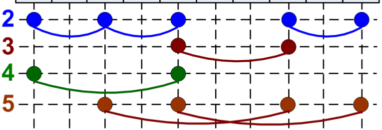
Core Periodic Concepts (CPC)

Core Periodic Concept

A periodic concept (I, P, T) is a **core periodic concept** if there does not exist any other periodic concept (I', P', T') such that $I = I'$, $P' \subset P$ and $T' \supset T$.

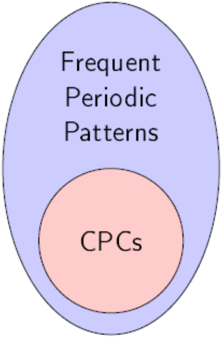
Core Periodic Concepts
$T_1(\{gF, dF\}, \{2\}, \{t_1, t_3, t_5, t_8, t_{10}\})$
$T_2(\{gF, dF\}, \{2, 4\}, \{t_1, t_5\})$
$T_3(\{gF, dF\}, \{2, 5\}, \{t_3, t_5, t_8, t_{10}\})$
$T_4(\{gF, dF\}, \{2, 3, 5\}, \{t_5, t_8\})$

t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
gF **dF**	I16 SI16	**gF** **dF**	w16 clk	**gF** **dF**	r16	fk gpid	**gF** **dF** pk	sup sd	**gF** **dF**



Condensed representation

- A condensed representation of a set S, is a subset C of the set S such that every element in S can be derived efficiently from C.
- Advantage: Less results.
- Algorithms to mine them are complex to design.
- Examples: closed [Pasquier et al., 1999], non-derivable [Calders & Goethals, 2002], etc.



How do we find CPCs?

3-STEP [Lopez Cueva et al., 2012]

1. Generate all triples.
2. Mine all periodic concepts using Data-Peeler [Cerf et al., 2009]
3. Extract CPCs.

Is there a more efficient way?

- *3-STEP* needs the whole set of periodic concepts to say whether a periodic concepts is a CPC.
- Connectivity property: only needs a periodic concept.

PerMiner: A CPC Miner

Can we directly enumerate CPCs?

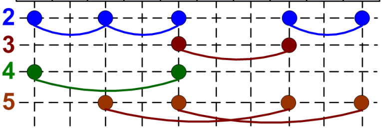
- There exist enumeration techniques to enumerate directly condensed representations [Uno et al., 2004] [Arimura & Uno, 2009].
- Polynomial delay time and polynomial space complexity.

PerMiner

- PerMiner bases its enumeration on these techniques: item enumeration with a special period handling.
- Depth-first search algorithm based on LCM [Uno et al., 2004] algorithm.
- Preserves polynomial delay time and polynomial space complexity (proven).
- Proven soundness, completeness and no duplicate generation.

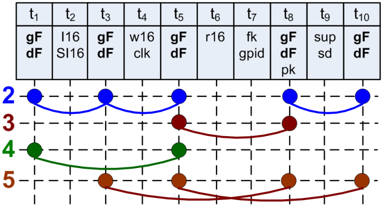
PerMiner: How does it work?

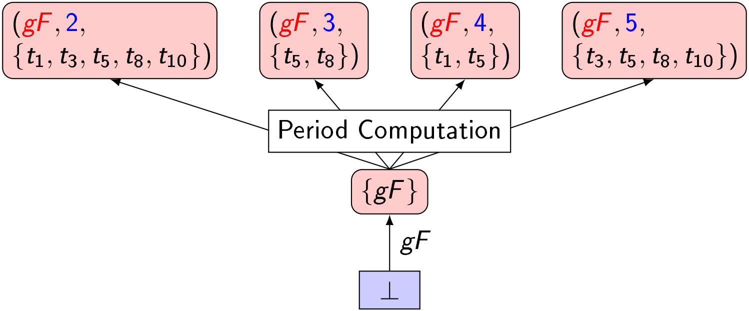
t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
gF **dF**	l16 SI16	**gF** **dF**	w16 clk	**gF** **dF**	r16	fk gpid	**gF** **dF** pk	sup sd	**gF** **dF**



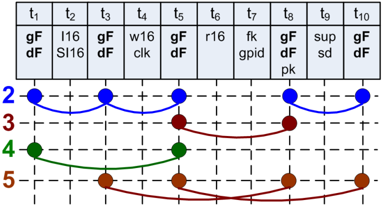
$$\begin{tikzcd} \{gF\} \\ \perp \arrow[u, "gF"'] \end{tikzcd}$$

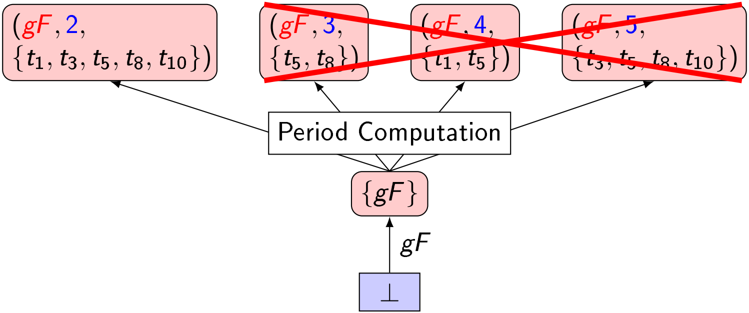
PerMiner: How does it work?





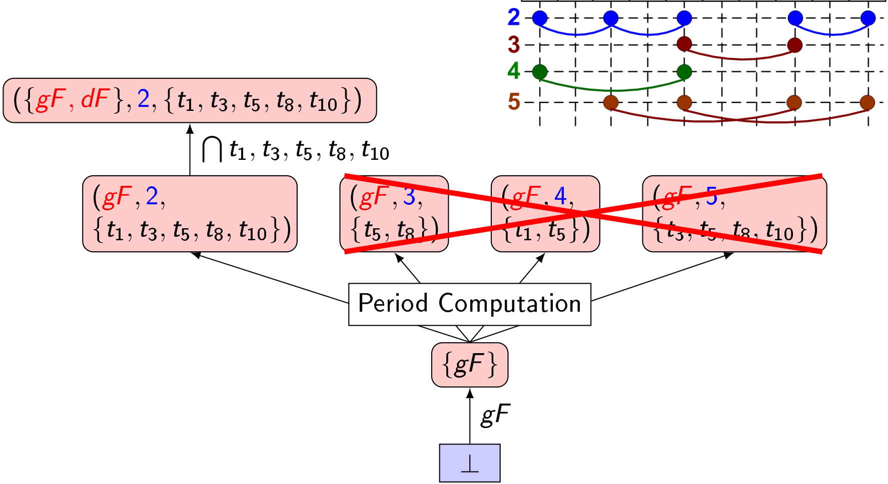
PerMiner: How does it work?



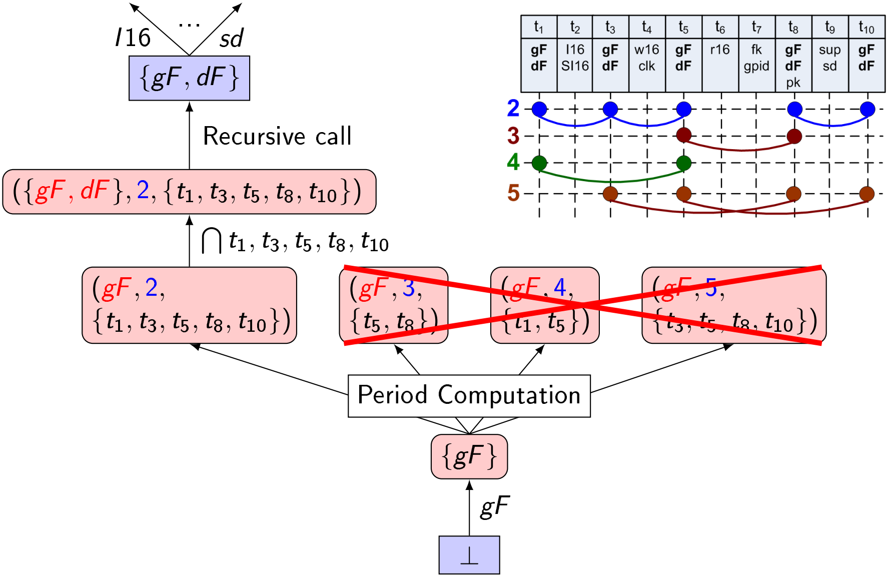


PerMiner: How does it work?

t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
gF **dF**	I16 SI16	**gF** **dF**	w16 clk	**gF** **dF**	r16	fk gpid	**gF** **dF** pk	sup sd	**gF** **dF**



PerMiner: How does it work?



t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
gF **dF**	l16 Sl16	**gF** **dF**	w16 clk	**gF** **dF**	r16	fk gpid	**gF** **dF** pk	sup sd	**gF** **dF**

Scalability Experiments

Experimental set-up

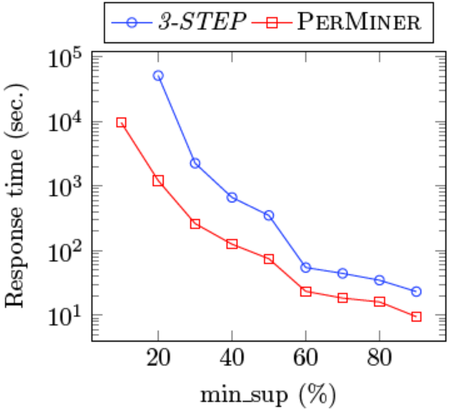
- PerMiner implemented in C++
- Run on a multiprocessor computing server:
  - 4 Intel Xeon X7560 processors (8 cores each) 2.27 GHz 64 GB RAM.

Real Data

- HNDTest Application: Test application for STMicroelectronics middleware for set-top boxes.
- Execution trace of a video playback: 528,360 events, 72 distinct events.
- Split into 10 ms intervals: 15,000 transactions, 35 items/transaction.

Scalability Experiments (Real Data)

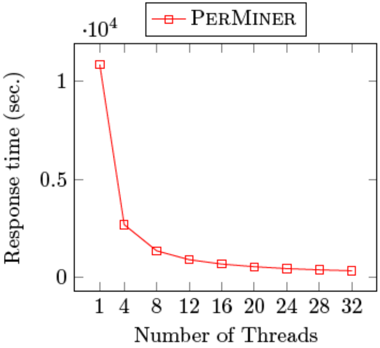
Experimental set-up: 1 core, 15,000 transactions, 35 items/transaction.

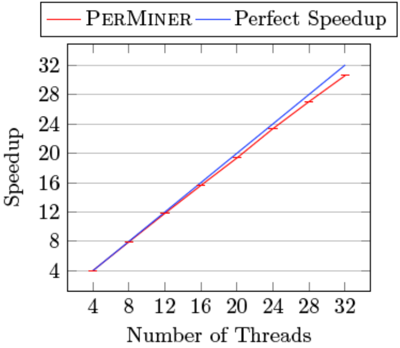


Scalability Experiments (Parallelization)

- Experimental set-up: 1 to 32 cores, minimum support 10%, 15,000 transactions, 35 items/transaction.

$$speedup_n = \frac{\textit{sequential execution time}}{\textit{execution time with n threads}}$$

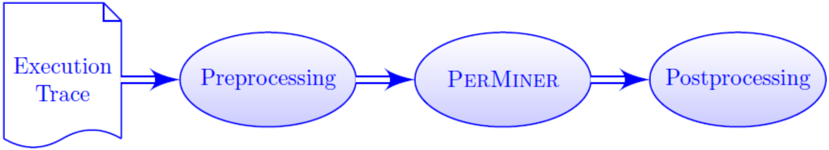




Outline

1. Introduction
   - Context and Motivation
2. Formal Framework
   - Frequent Periodic Patterns
   - Core Periodic Concepts
   - PerMiner Algorithm
   - Scalability Experiments
3. Its Application
   - Usage Guidelines
   - Use Cases
4. Conclusions
   - Conclusions
   - Future Work

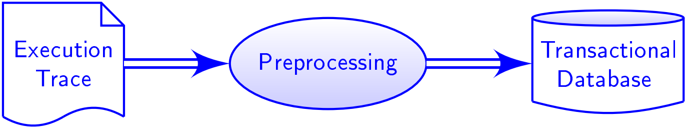
Usage Guidelines



Special case of KDD methodology

- Software developers are not familiar with data mining techniques.
- A methodology gives developers the necessary guidelines to exploit this new technique.

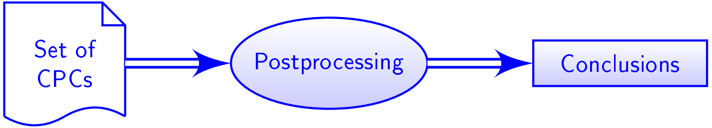
Preprocessing



Consists in transforming an execution trace into a transactional database by splitting the trace in a sequence of sets of elements

- Which information is important?
  - *GetFrame*, 135_*GetFrame*, *GetFrame_OK*, ...
- Which splitting criterion better suits the required analysis?
  - We proposed two methods: Time interval and function name.
  - Domain specific knowledge might propose better suited methods.

Postprocessing



How to analyze PERMINER results?

- Manually is not manageable.
- Visualization and analysis tools are needed.
- We have implemented two tools:
  - Analysis tool: Competitors Finder.
  - Visualization tool: *CPCViewer*.

Competitors Finder

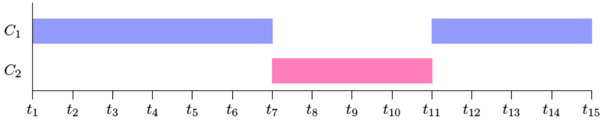
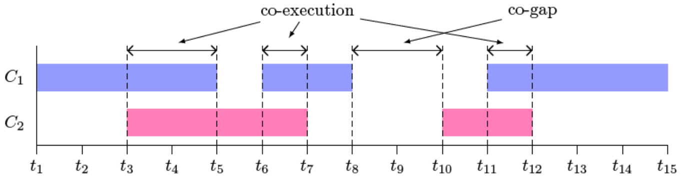


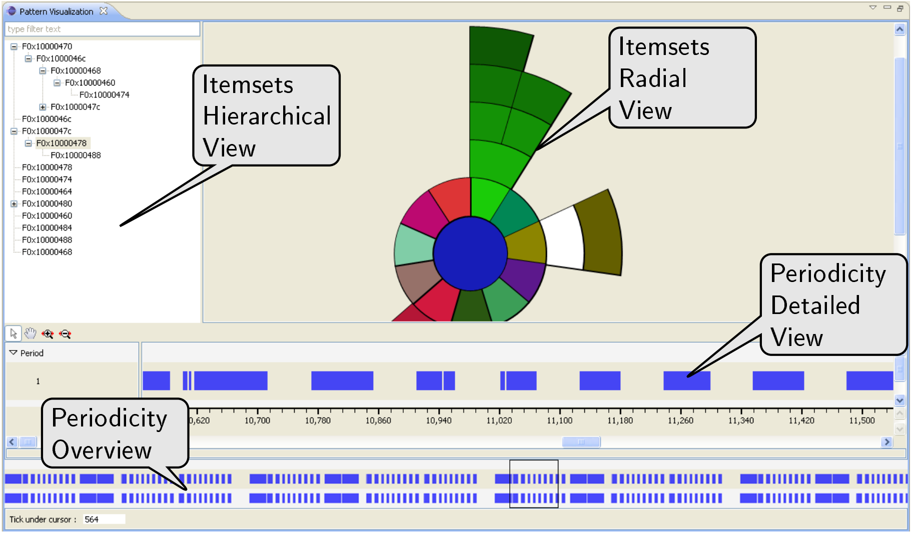
Figure: Example of two patterns in full competition

Calculating Competition Ratio

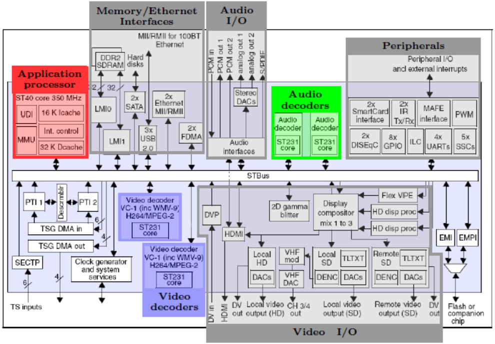


- Competition Ratio = 100 % - (co-execution + co-gap)
- If Competition Ratio $\geq$ Minimum Competition Ratio $\Rightarrow$ Competitors!

CPCViewer


Pattern Visualization
type filter text
F0x10000470
F0x1000046c
F0x10000468
F0x10000460
F0x10000474
F0x1000047c
F0x1000046c
F0x1000047c
F0x10000478
F0x10000488
F0x10000478
F0x10000474
F0x10000464
F0x10000480
F0x10000460
F0x10000484
F0x10000488
F0x10000468
Itemsets Hierarchical View
Itemsets Radial View
Periodicity Detailed View
Period
1
10,700
10,780
10,860
10,940
11,020
11,100
11,180
11,260
11,340
11,420
11,500
Periodicity Overview
Tick under cursor : 564

1st Use Case: HNDTest Application



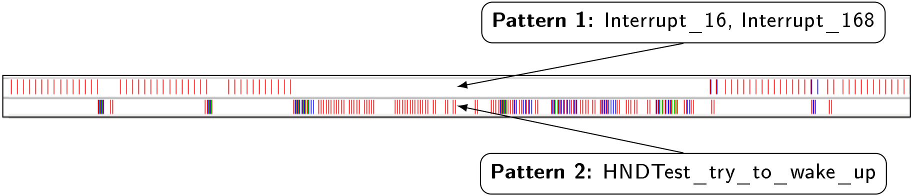
- STAPI: set-top box middleware.
- STi7200: set-top box SoC.
- Tracing: *KPTrace* kernel module.
- Trace split into 1 ms intervals.
- PerMiner (10%): 758 CPCs in 195 s.

Pattern	Quantity
FPP	18,459
PC	51,446
CPC	758

1st Use Case: HNDTest Application

Discovered conflict between the application and the system (USB port)

- `Interrupt_16`: processor clock interrupt.
- `Interrupt_168`: USB interrupt.
- `HNDTest_try_to_wake_up`: system call (`try_to_wake_up`).



2nd Use Case: GStreamer Application

Multimedia application using GStreamer multimedia framework.

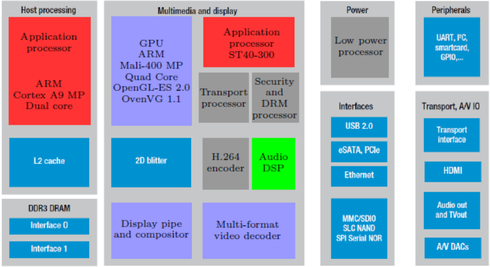
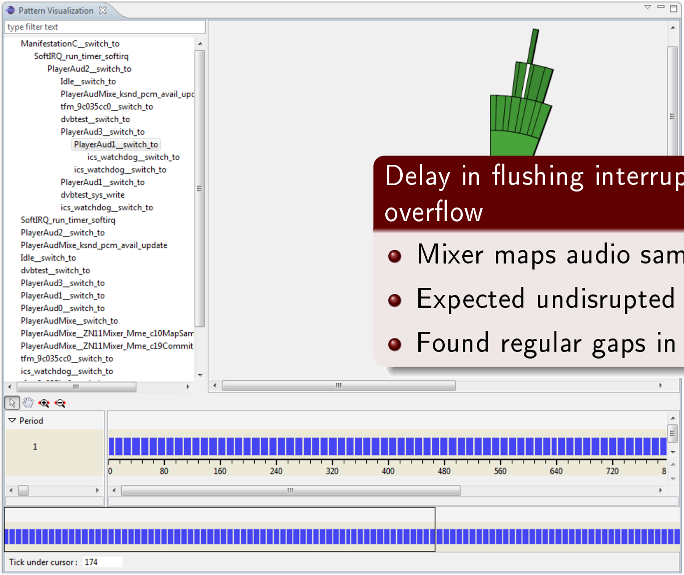


Figure: Orly SoC Block Diagram

- Orly STiH416 multi-core MPSoC.
- Tracing: *KPTrace* kernel module.
- Trace split into 32 ms intervals.
- PerMiner (10%): 787 CPCs in 28 s.

Pattern	Quantity
FPP	3,086,321
PC	21,588
CPC	787

2nd Use Case: GStreamer Application



Delay in flushing interrupt causing buffer overflow

- Mixer maps audio samples every 32ms.
- Expected undisrupted pattern of period 1.
- Found regular gaps in periodicity.

Outline

1. Introduction
   - Context and Motivation
2. Formal Framework
   - Frequent Periodic Patterns
   - Core Periodic Concepts
   - PerMiner Algorithm
   - Scalability Experiments
3. Its Application
   - Usage Guidelines
   - Use Cases
4. Conclusions
   - Conclusions
   - Future Work

Conclusions

Objective

- Help developers in debugging multimedia embedded applications.
- Proposed a new approach that makes use of periodic pattern mining.

Pattern Mining

- Defined condensed representation of frequent periodic patterns: Core Periodic Concepts (CPC).
- Implemented efficient CPC miner algorithm PERMINER.

Embedded Systems

- Given some guidelines to use our approach.
- Developed postprocessing tools: *CPCViewer* and Competitors Finder.

Future Work

Pattern Mining

- Define enumeration strategy based on items and periods.
- Explore different types of periodic patterns: sequences, graphs, etc.
- Include domain knowledge in the mining process (SoCTrace, Leon Fopa PhD).

Analysis

- Automatic detection of anomalies.
- Definition of a full methodology (CIFRE, Oleg Iegorov PhD).

Questions?

Triadic Approach

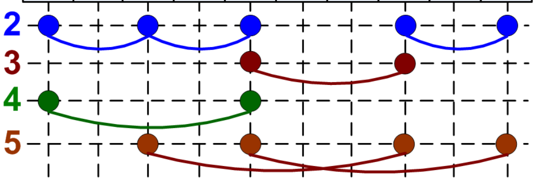
Periodic Triadic Context $(\mathcal{I}, \mathcal{P}, \mathcal{T}, \mathcal{Y})$

- $\mathcal{I}$ set of items.
- $\mathcal{P}$ set of periods.
- $\mathcal{T}$ set of transactions.
- $\mathcal{Y}$ ternary relation, $\mathcal{Y} \subseteq \mathcal{I} \times \mathcal{P} \times \mathcal{T}$.

Example (min_sup = 2)

- $\mathcal{I} = \{gF, dF, I16, SI16, ...\}$
- $\mathcal{P} = \{1..5(|D|/min\_sup)\}$
- $\mathcal{T} = \{t_1, ..., t_{10}\}$
- $\mathcal{Y} = \{(gF, 2, t_1), (gF, 2, t_3), (gF, 2, t_5), (gF, 2, t_8), (gF, 2, t_{10}), (dF, 2, t_1), (dF, 2, t_3), (dF, 2, t_5), ...\}$

t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
gF **dF**	I16 SI16	**gF** **dF**	w16 clk	**gF** **dF**	r16	fk gpid	**gF** **dF** pk	sup sd	**gF** **dF**



Triadic Approach

Periodic Concept

- A triple (I, P, T) is **frequent** if $I \neq \emptyset$, $P \neq \emptyset$ and $|T| \geq min\_sup$.
- A frequent triple (I, P, T) is a **periodic concept** if none of its three components can be enlarged without violating the condition $I \times P \times T \subseteq \mathcal{Y}$.
  Example: $T_1(\{gF, dF\}, \{2\}, \{t_1, t_3, t_5, t_8, t_{10}\})$.

	Periods	2										3									
Itemsets	Transactions	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
gF		X		X		X			X		X					X			X		
dF		X		X		X			X		X					X			X		
...																					

	Periods	4										5									
Itemsets	Transactions	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
gF		X				X								X		X			X		X
dF		X				X								X		X			X		X
...																					

Connectivity Properties of Core Periodic Concepts

Definition

tidlist(X, p) returns the list of all t transactions such that $(X, p, t) \in \mathcal{Y}$.

Theorem

A periodic concept (X, P, T) is a core periodic concept if and only if for all $p \in P$ it is true that *tidlist*$(X, p) = T$.

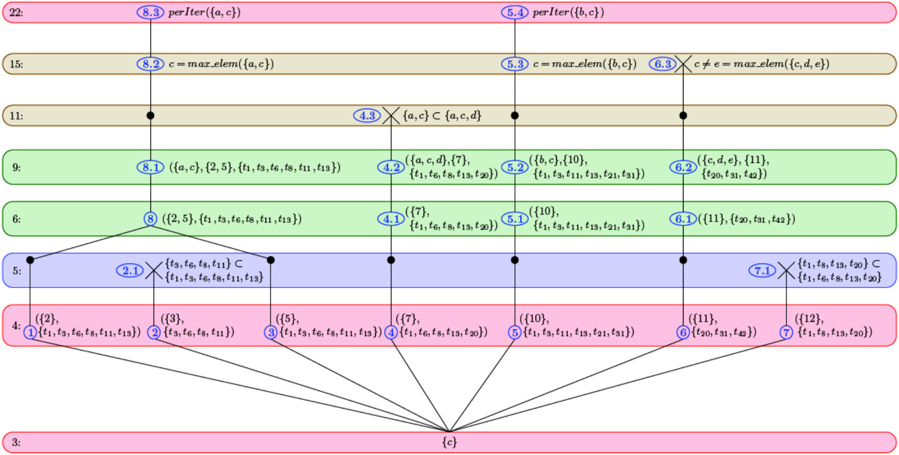
Definition

A triple (X, P, T) is fully connected if $\forall p \in P$ and $\forall t \in T$ there exist $t' \in T \setminus \{t\}$ such that the distance between t and t' is equal to p.

Proposition

A core periodic concept is fully connected.

t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14-19}	t_{20}	t_{21}	t_{22-30}	t_{31}	t_{32-41}	t_{42}
a	.	a	.	.	a	.	a	.	.	a	.	a	.	a	b	.	b	.	c
b	.	b	.	.	c	.	c	.	.	b	.	b	.	c	c	.	c	.	d
c	.	c	.	.	d	.	d	.	.	c	.	c	.	d	.	.	d	.	e
d	.	.	.	.	.	.	.	.	.	.	.	d	.	e	.	.	e	.	.



procedure PerMiner (D, min_sup);
Data: dataset D, minimum support threshold min_sup
Result: Output all Core Periodic Concepts that occur in D
begin
    if $|D| \geq min\_sup$ **then**
        $\bot_{clo} \leftarrow \bigcap_{t \in D} t$
        output $(\bot_{clo}, \{1..|D|/2\}, D)$
        $D_{\bot_{clo}} = \{t \setminus \bot_{clo} | t \in D\}$
        foreach $e \in \mathcal{I}$ *with* $e \notin \bot_{clo}$ **do**
            $perIter(\bot_{clo}, D_{\bot_{clo}}, e, \emptyset, min\_sup)$

```
procedure perIter(X, D_X, e, el, min_sup);
Data: Itemset of a discovered CPC X, reduced dataset D_X, item e, exclusion list el, minimum support threshold
      min_sup.
Result: Output all Core Periodic Concepts whose itemset is prefixed by X and whose transactions are in D_X, with
        minimal support min_sup.
begin
    A := {e}
    B := getPeriods(tidlist(A), min_sup)                                  /* Period computation */
    B' := B \ {b | ∃b' ∈ B such that b.occs ⊂ b'.occs}
    G := group(B')
    S ← ∅                                                                 /* Closure computation */
    foreach g ∈ G do
        A' := ⋂_{t ∈ g.occs} t
        S := S ∪ (A', g.periods, g.occs)
    S := filter(S);                                                       /* First parent test */
    new_el ← el
    enum ← ∅                                                              /* Itemset enumeration */
    foreach (A', P, T) ∈ S do
        if max_elem(A') = e then
            Q = X ∪ A'
            if el_test(Q, el) then
                output (Q, P, T)
                if Q ∉ enum then
                    D_Q = reduce(D_X, Q, e, min_sup)                      /* Dataset Reduction */
                    foreach i ∈ 𝓘 with i < e and i ∉ Q do
                        perIter(Q, D_Q, i, new_el, min_sup)
                    enum := enum ∪ Q
                new_el := new_el ∪ Q
```

```
function getPeriods(T, min_sup)
Data: Transaction list T, minimum support threshold min_sup
Result: A list of tuples (period, transaction list of the period)
B ← ∅
foreach period ∈ [1..|D|/min_sup] do
    b.occs ← ∅
    b.periods := period
    i := 0
    while i < (|T| − 1) do
        if T[i].checked == false then
            j := i + 1
            while j < |T| AND (T[j] − T[i]) <= period do
                if (T[j] − T[i]) == period then
                    b.occs := b.occs ∪ i; T[i].checked := true
                    b.occs := b.occs ∪ j; T[j].checked := true
                    k := j + 1
                    while k < |T| AND (T[k] − T[j]) <= period do
                        if (T[k] − T[j]) == period then
                            b.occs := b.occs ∪ k; T[k].checked := true
                            j := k
                        k++
                j++
        i++
    if |b.occs| >= min_sup then
        B := B ∪ b
return B
end function
```

function *group*(B)

Data: List of tuples (period, transaction list of the period) B

Result: A list of tuples grouped by transaction list

foreach $b, b' \in B$ **do**

  if $b.occs == b'.occs$ **then**

    $b.periods := b.periods \cup b'.periods$

    $B := B \setminus b'$

return B

end function

```
function el_test(Q, el)
  Data: Itemset Q, Exclusion list el
  Result: True if none of the elements in el is included in Q. False
          otherwise.
foreach X ∈ el do
    if X ⊂ Q then
        return False
return True
end function
```

function *filter*(S)
Data: List of CPCs S
Result: A filtered list of CPCs
foreach $(A, P, C), (A', P', C') \in S$ **do**
    if $A \subset A'$ **then**
        $S := S \setminus (A', P', C')$
return S
end function

function *reduce*($D_X^{reduced}, A', e, min\_sup$)

Data: Database $D_X^{reduced}$, Itemset A', element e, minimum support threshold min_sup

Result: Reduced Database of A': $D_{A'}^{reduced}$

$D_{A'}^{reduced} = D_X^{reduced}[e]$

foreach $i \in \mathcal{I}$ **do** `/* All items of` $\mathcal{I}$ `with support smaller than */`

`/*` min_sup `are removed from the database */`

  if $support(i) < min\_sup$ **then**

    Suppress i from all transactions in $D_{A'}^{reduced}$

foreach $i \in A'$ **do** `/* All items of` A' `are removed from the database */`

  Suppress i from all transactions in $D_{A'}^{reduced}$

return $D_{A'}^{reduced}$

end function

Why not a closure operator?

- Closed Patterns:
  - Offer a reduced representation of the set of frequent patterns.
  - but **closure operators** based on binary relations.
- Periodic Patterns:
  - Ternary relation: the item **i** is found in the transaction **t** with period **p**.
- Closure operator for n-ary relations with $n > 2$.
  - "It is no longer possible to enumerate one attribute domain (usually items) and compute the rest of the pattern thanks to a Galois connection" [CERF09].