Partially Safe Evolution of SPLs

Gabriela Cunha Sampaio

Supervisor: Paulo Henrique Monteiro Borba

Co-supervisor: Leopoldo Motta Teixeira

March 27, 2017

http://www.bmwmainline.com/bmw-genius

http://www.macworld.co.uk/review/iphone/iphone-se-vs-iphone-6s-comparison-review-3637407/

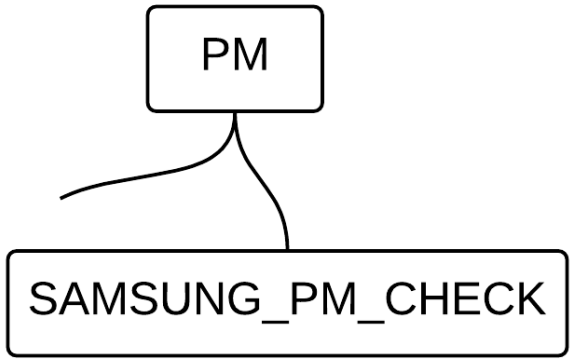
https://www.qt.io/case-built-with-qt-ubuntu-open-source-software-platform/

Advantages

- Tailor-made
- Reduced costs
- Improved quality
- Time to market

SPL Elements

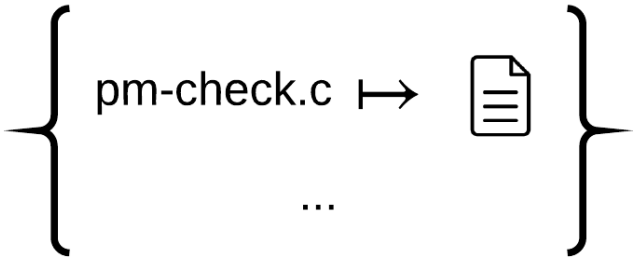
FM



Configuration Knowledge

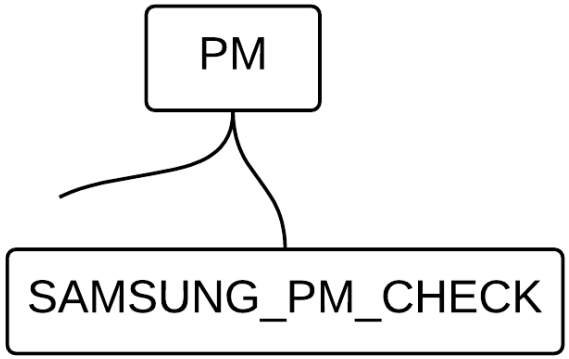
CONFIG_SAMSUNG_PM_CHECK	pm-check.o
...	...

Asset Mapping



SPL Elements

FM



Kconfig

```
config SAMSUNG_PM_CHECK
bool "S3C2410 PM Suspend
Memory CRC"
depends on PM
select CRC32
...
```

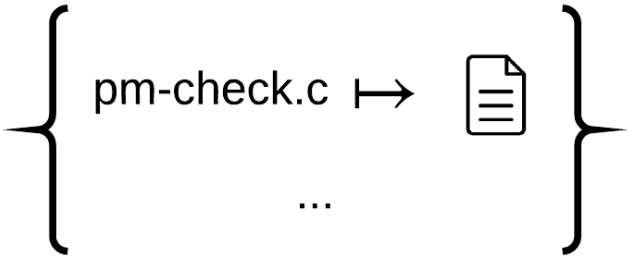
Configuration Knowledge

CONFIG_SAMSUNG_PM_CHECK	pm-check.o
...	...

Makefile

```
obj-$(CONFIG_SAMSUNG_
PM_CHECK) += pm-check.o
...
```

Asset Mapping

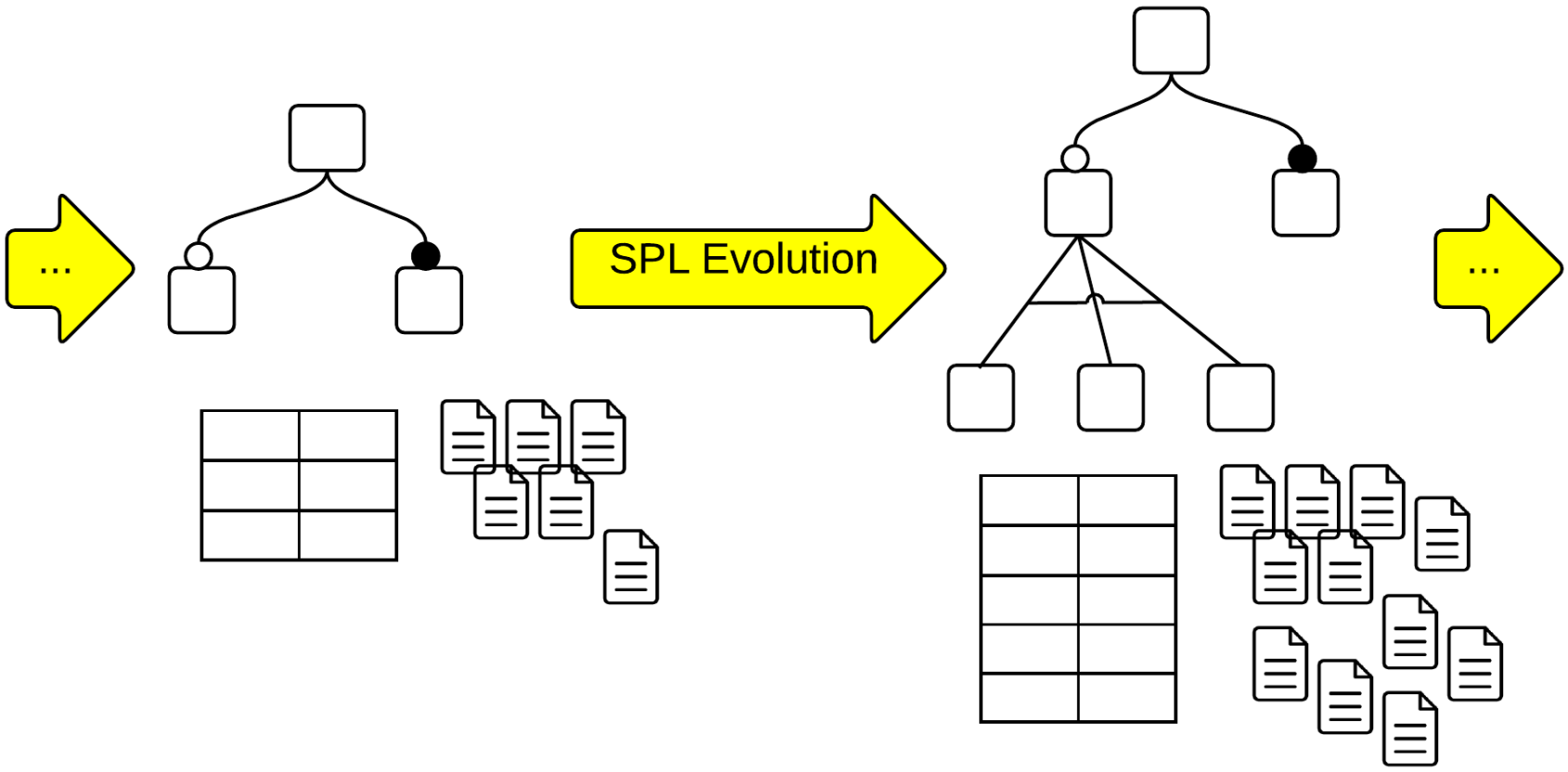


Implementation

```
#ifdef
    CONFIG_SAMSUNG_PM_CHECK
    ...
```

pm-check.c

Software Product Line Evolution



Feature Renaming Scenario

drivers/gpio/Kconfig

```
- config GPIO_LANGWELL
+ config GPIO_INTEL_MID
```

drivers/gpio/Makefile

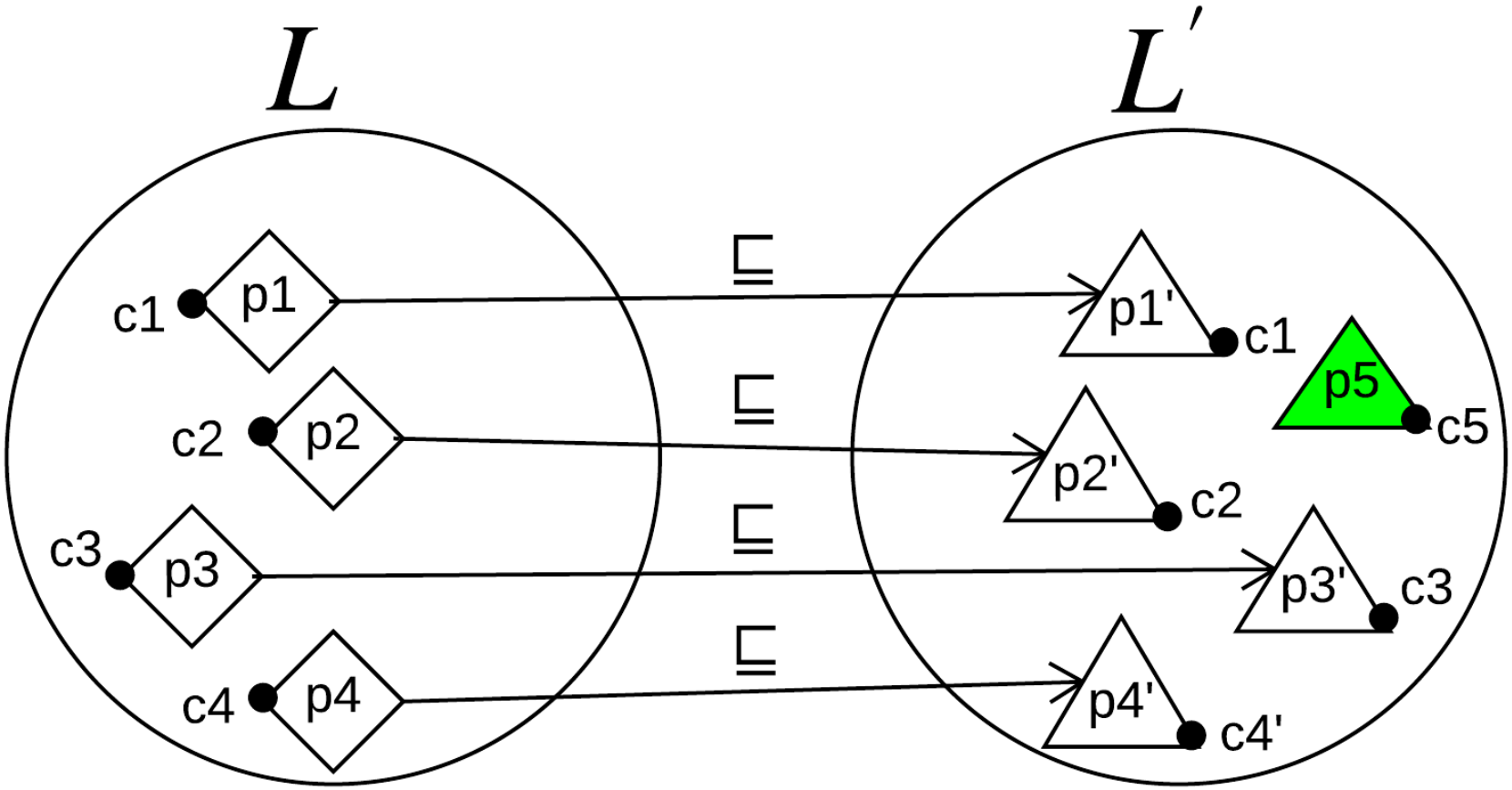
```
- obj-$( CONFIG_GPIO_LANGWELL ) += gpio- langwell .o
+ obj-$( CONFIG_GPIO_INTEL_MID ) += gpio- intel-mid .o
```

drivers/gpio/gpio-langwell.c → drivers/gpio/gpio-intel-mid.c

File renamed without changes.

Commit 84743ea369

Safe Evolution

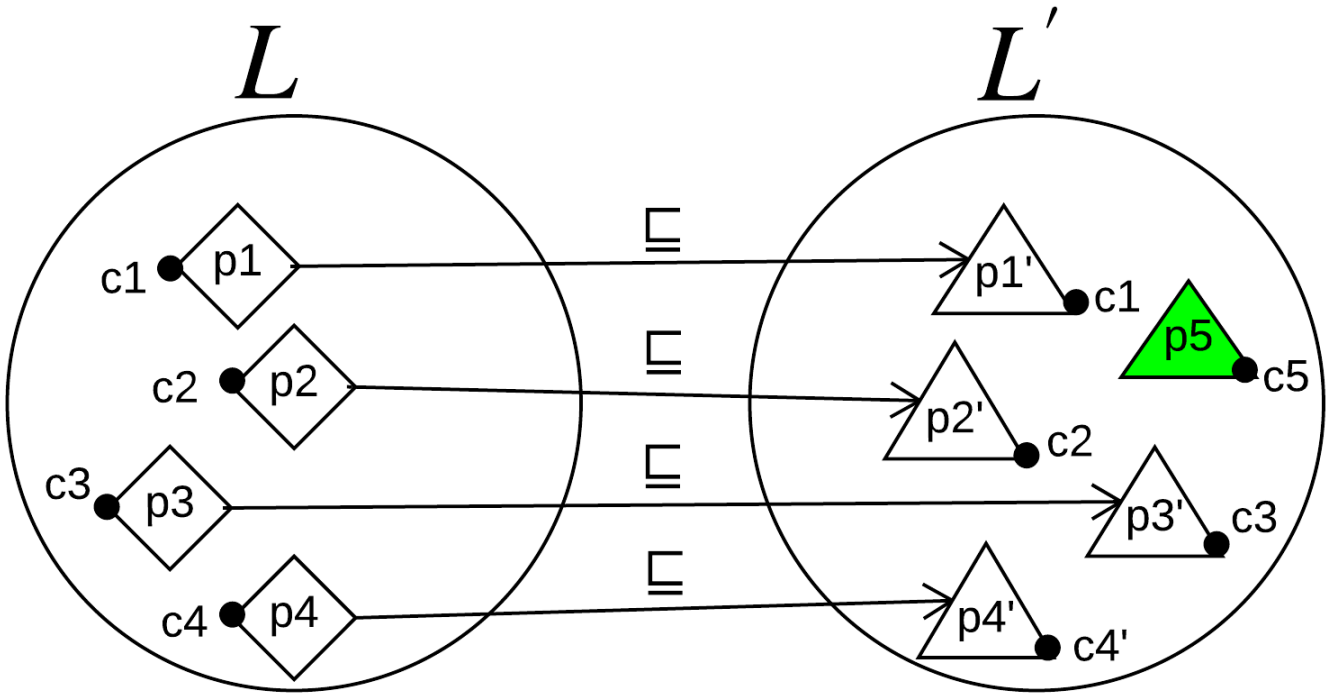


Every product from *L* has a *compatible* product in *L'*

SPL Refinement

$L \sqsubseteq L'$ whenever

$$\forall c \in [\![F(L)]\!] \cdot \exists c' \in [\![F(L')]\!] \cdot prod(L, c) \sqsubseteq prod(L', c')$$



Feature Removal Scenario

12 drivers/leds/Kconfig

```
- config LEDS_RENESAS_TPU
-       bool "LED support for Renesas TPU"
-       depends on LEDS_CLASS=y && HAVE_CLK && GPIOLIB
-       help
-           ...
```

1 drivers/leds/Makefile

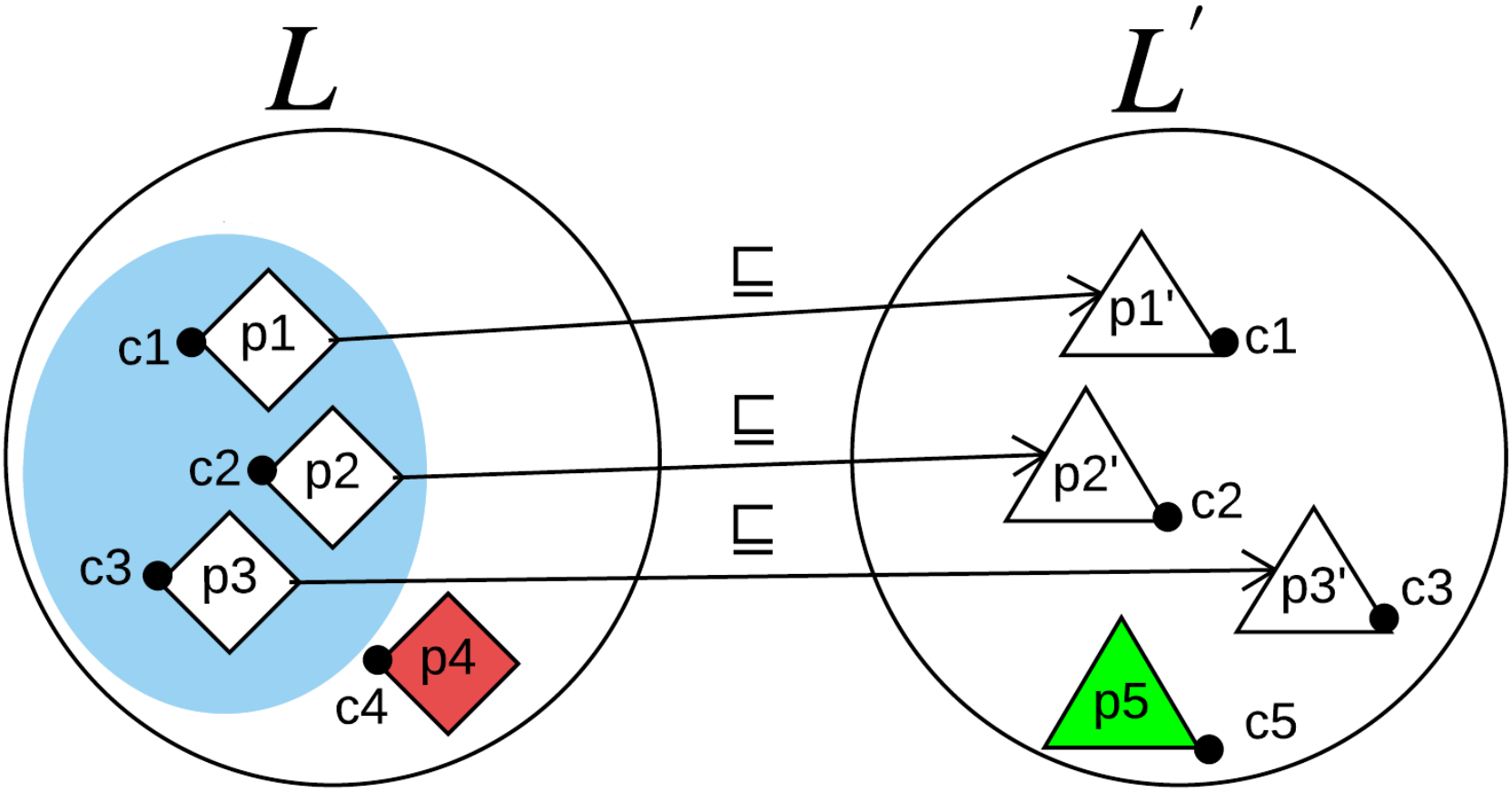
```
- obj-$(CONFIG_LEDS_RENESAS_TPU) += leds-renesas-tpu.o
```

337 drivers/leds/leds-renesas-tpu.c

14 include/linux/platform_data/leds-renesas-tpu.h

Commit ae3e4c2776

Partially Safe Evolution



There is no product compatible with *p*4 in *L'*

Objectives

- Define partial refinement theory
- Derive properties
- Provide transformation patterns
- Evaluate these patterns

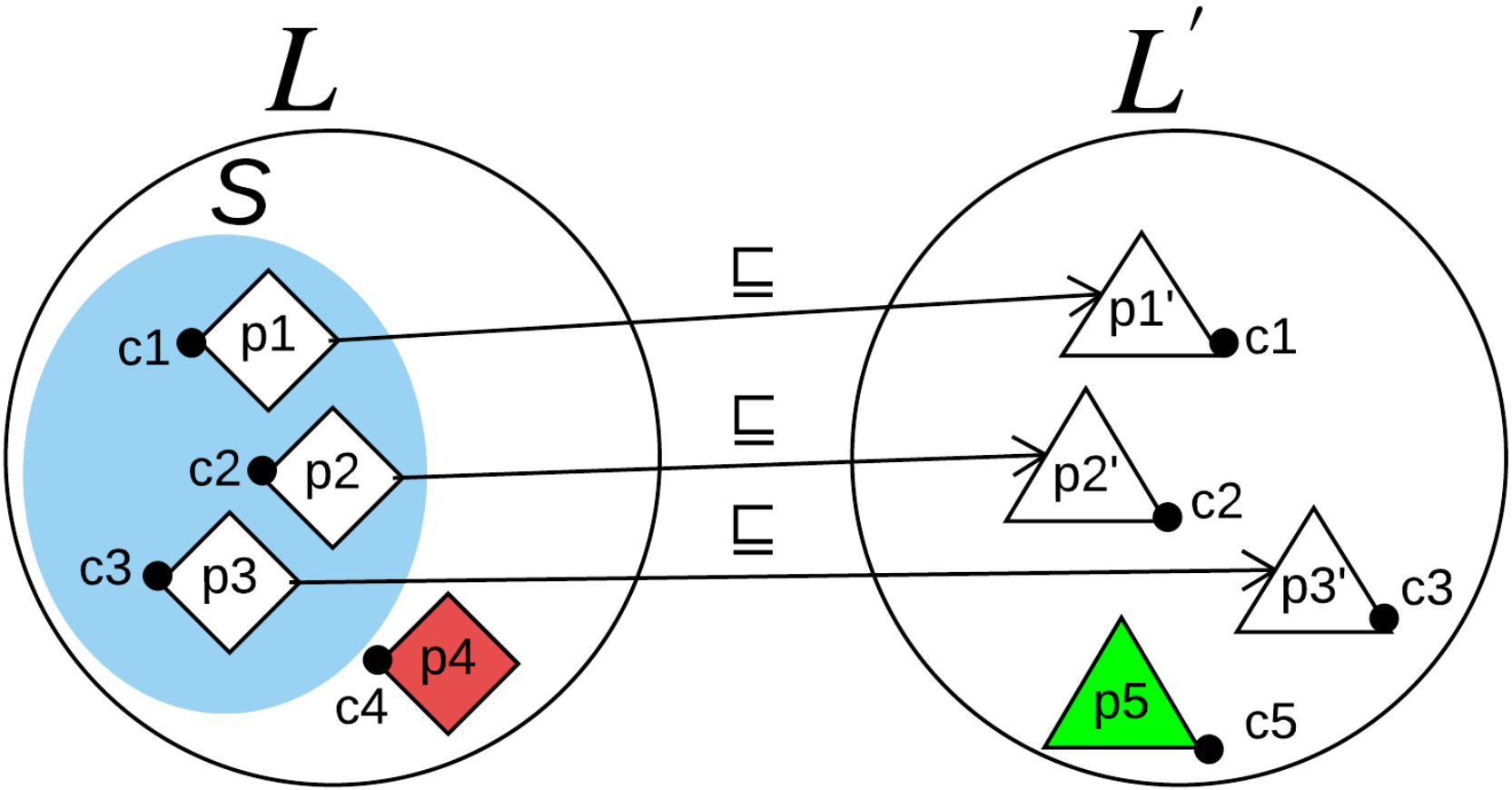
Refinement x Partial Refinement

$$L \sqsubseteq L' \qquad\qquad L \sqsubseteq_S L'$$

S is a set of configurations

Defining SPL Partial Refinement

$$L \sqsubseteq_S L'$$



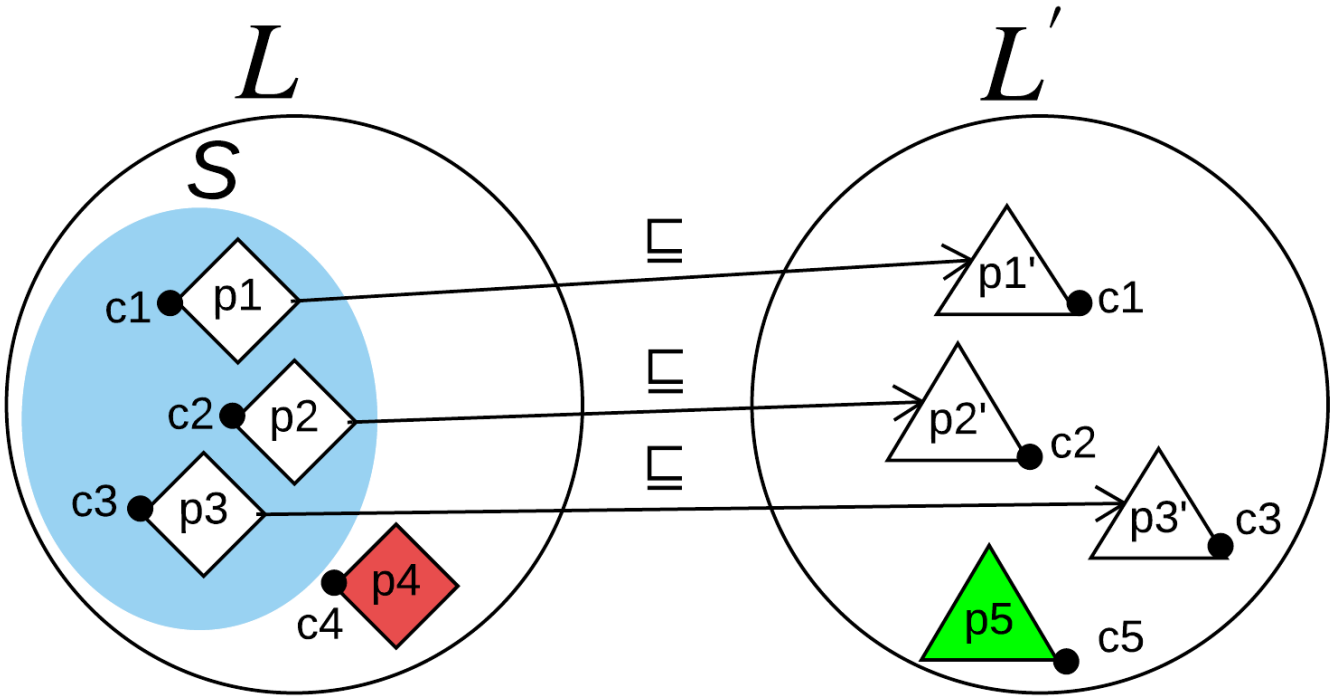
SPL Partial Refinement

$L \sqsubseteq_S L'$ whenever

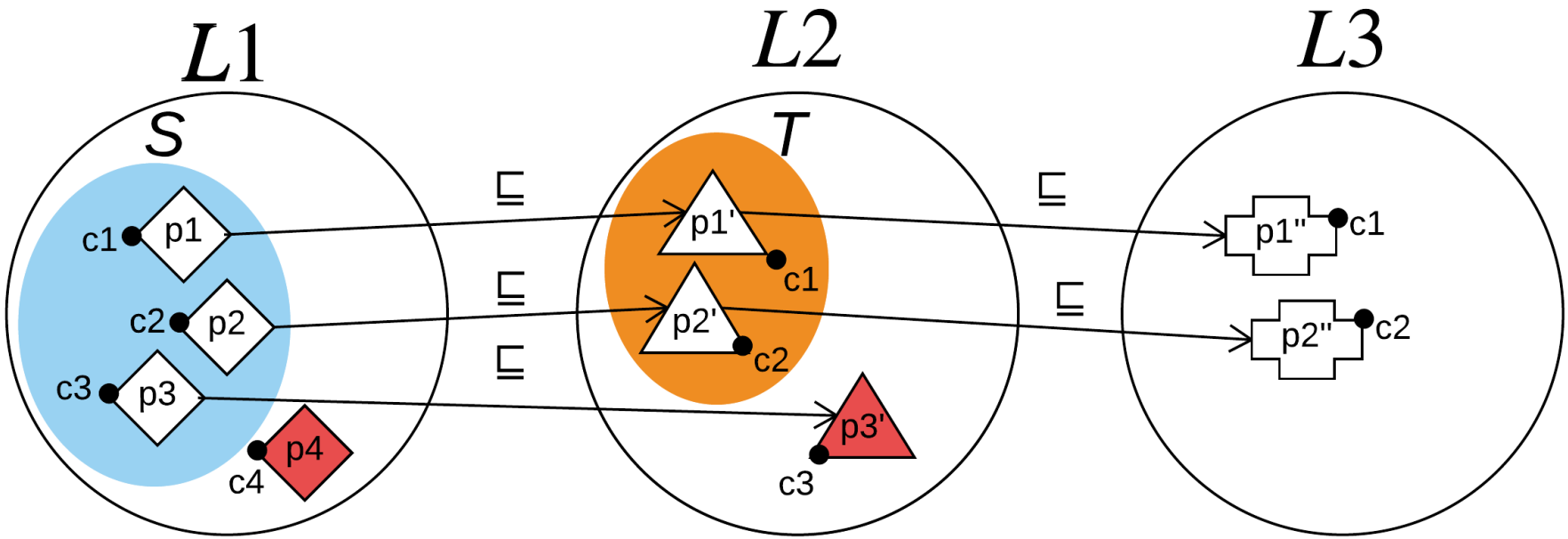
$$S \subseteq [\![F(L)]\!] \wedge S \subseteq [\![F(L')]\!]$$

and

$$\forall c \in S \cdot prod(L, c) \sqsubseteq prod(L', c)$$



Partial Refinement is a Pre-order



$$L1 \sqsubseteq_S L2 \wedge L2 \sqsubseteq_T L3$$
$$\Rightarrow$$
$$L1 \sqsubseteq_{S \cap T} L3$$

Refinement is not particular case of Partial Refinement

If $S = [\![F(L)]\!]$, $L \sqsubseteq L' \neq L \sqsubseteq_S L'$

drivers/gpio/Kconfig

```
- config GPIO_LANGWELL
+ config GPIO_INTEL_MID
```

drivers/gpio/Makefile

```
- obj-$( CONFIG_GPIO_LANGWELL ) += gpio- langwell .o
+ obj-$( CONFIG_GPIO_INTEL_MID ) += gpio- intel-mid .o
```

drivers/gpio/gpio-langwell.c → drivers/gpio/gpio-intel-mid.c

File renamed without changes.

Commit 84743ea369

In this example, $L \sqsubseteq L'$, but $L \not\sqsubseteq_S L'$

Weak Partial Refinement

$L \sqsubseteq_f L'$ whenever

$$\forall c \in dom(f) \cdot f(c) \in [\![F(L')]\!] \wedge prod(L, c) \sqsubseteq prod(L', f(c))$$

Four Refinement Notions

- SPL Refinement - $L \sqsubseteq L'$
- Name Aware SPL Refinement - $L \preceq L'$
- Partial SPL Refinement - $L \sqsubseteq_S L'$
- Weak Partial SPL Refinement - $L \sqsubseteq_f L'$

Combining Safe and Partially Safe Situations

- Safe and partially safe notions can be used together
- Partial and full refinement are applied interchangeably
- We should provide support in such situations
- We define properties that guarantee refinement after two different types of evolution scenarios

Partially Safe Evolution Scenario - Feature Removal

12 drivers/leds/Kconfig

```
- config LEDS_RENESAS_TPU
-       bool "LED support for Renesas TPU"
-       depends on LEDS_CLASS=y && HAVE_CLK && GPIOLIB
-       help
-           ...
```

1 drivers/leds/Makefile

```
- obj-$(CONFIG_LEDS_RENESAS_TPU) += leds-renesas-tpu.o
```

337 drivers/leds/leds-renesas-tpu.c

14 include/linux/platform_data/leds-renesas-tpu.h

Commit ae3e4c2776

Safe Evolution Scenario - Feature Renaming

drivers/gpio/Kconfig

```
- config GPIO_LANGWELL
+ config GPIO_INTEL_MID
```

drivers/gpio/Makefile

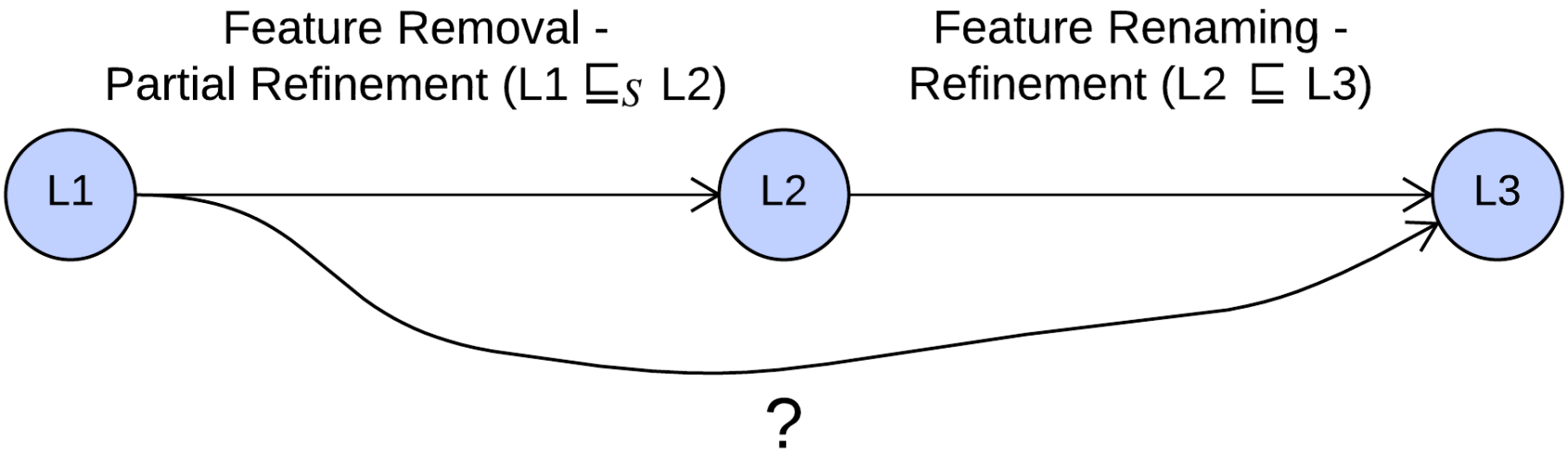
```
- obj-$( CONFIG_GPIO_LANGWELL ) += gpio- langwell .o
+ obj-$( CONFIG_GPIO_INTEL_MID ) += gpio- intel-mid .o
```

drivers/gpio/gpio-langwell.c → drivers/gpio/gpio-intel-mid.c

File renamed without changes.

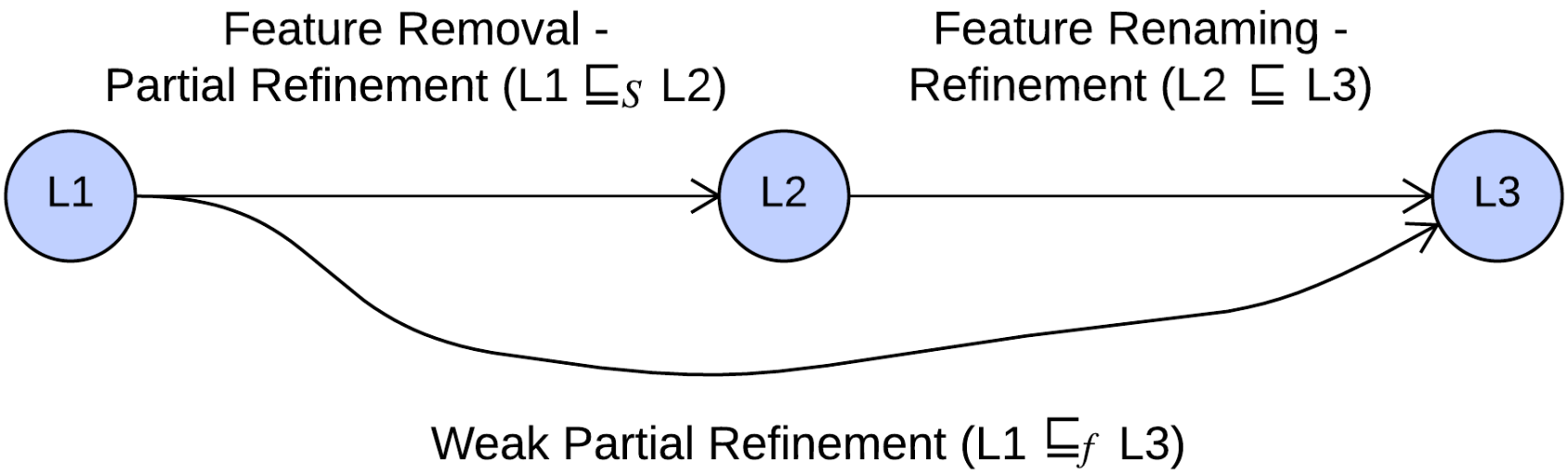
Commit 84743ea369

Feature Removal and Renaming



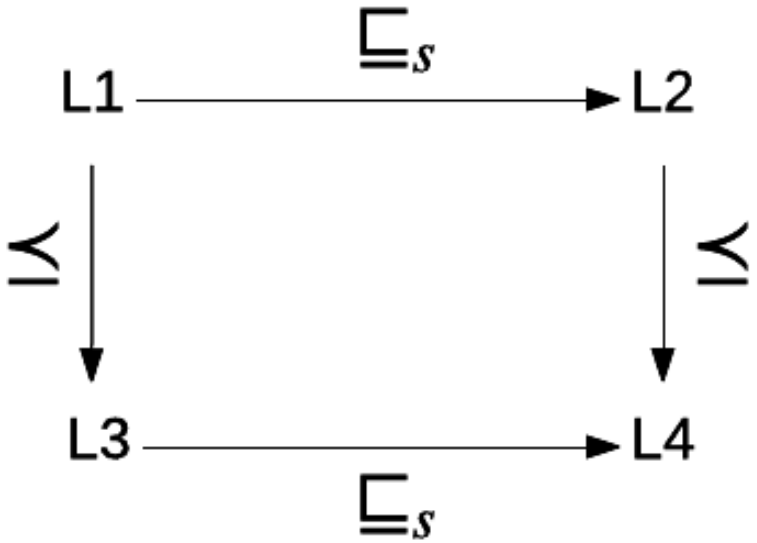
S is the set of configurations that do not have *LEDS_RENESAS_TPU*

The result is a Weak Partial Refinement

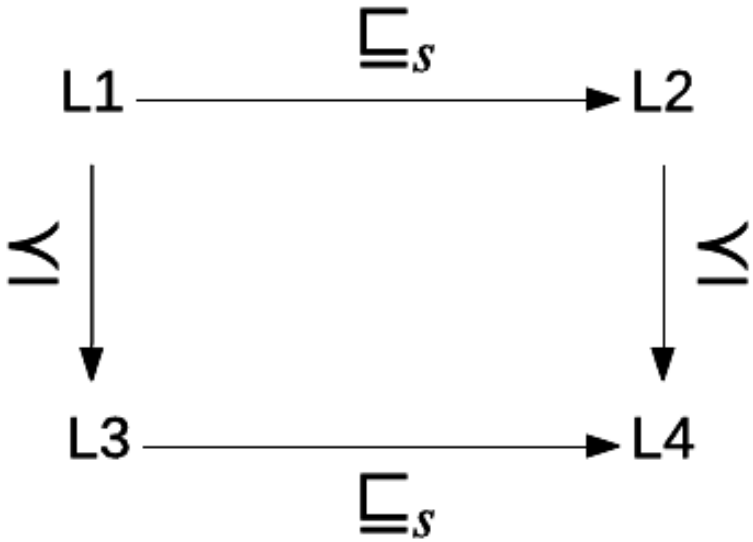


$$f(c) = c[GPIO\_LANGWELL / GPIO\_INTEL\_MID]$$

Commutativity of Refinement and Partial Refinement



Commutativity of Refinement and Partial Refinement



$$\forall L1, L2, L4, S \cdot L1 \sqsubseteq_S L2 \wedge L2 \preceq L4 \Rightarrow \exists L3 \cdot L1 \preceq L3 \wedge L3 \sqsubseteq_S L4$$

$$\forall L1, L3, L4, S \cdot S \subseteq [\![F(L1)]\!] \wedge L1 \preceq L3 \wedge L3 \sqsubseteq_S L4 \Rightarrow \exists L2 \cdot L1 \sqsubseteq_S L2 \wedge L2 \preceq L4$$

Partially Safe Evolution Templates

- Abstractions of partially safe evolution scenarios
- Defined based on preexisting templates
- Precisely determine which subset of products is refined for each situation
- Templates are easier to understand comparing to the theory

Remove Feature Example

12 drivers/leds/Kconfig

```
- config LEDS_RENESAS_TPU
-       bool "LED support for Renesas TPU"
-       depends on LEDS_CLASS=y && HAVE_CLK && GPIOLIB
-       help
-           ...
```

1 drivers/leds/Makefile

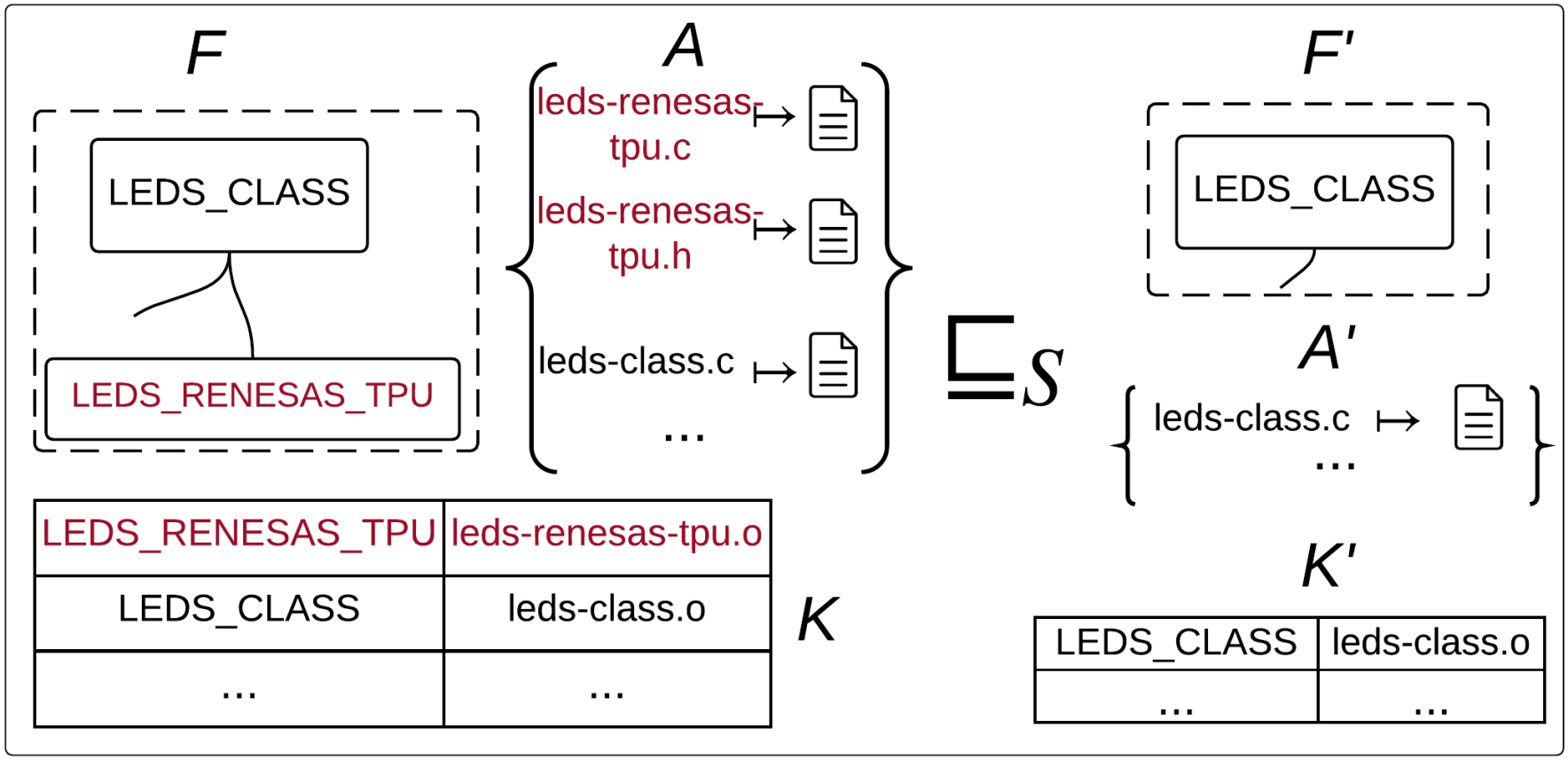
```
- obj-$(CONFIG_LEDS_RENESAS_TPU) += leds-renesas-tpu.o
```

337 drivers/leds/leds-renesas-tpu.c

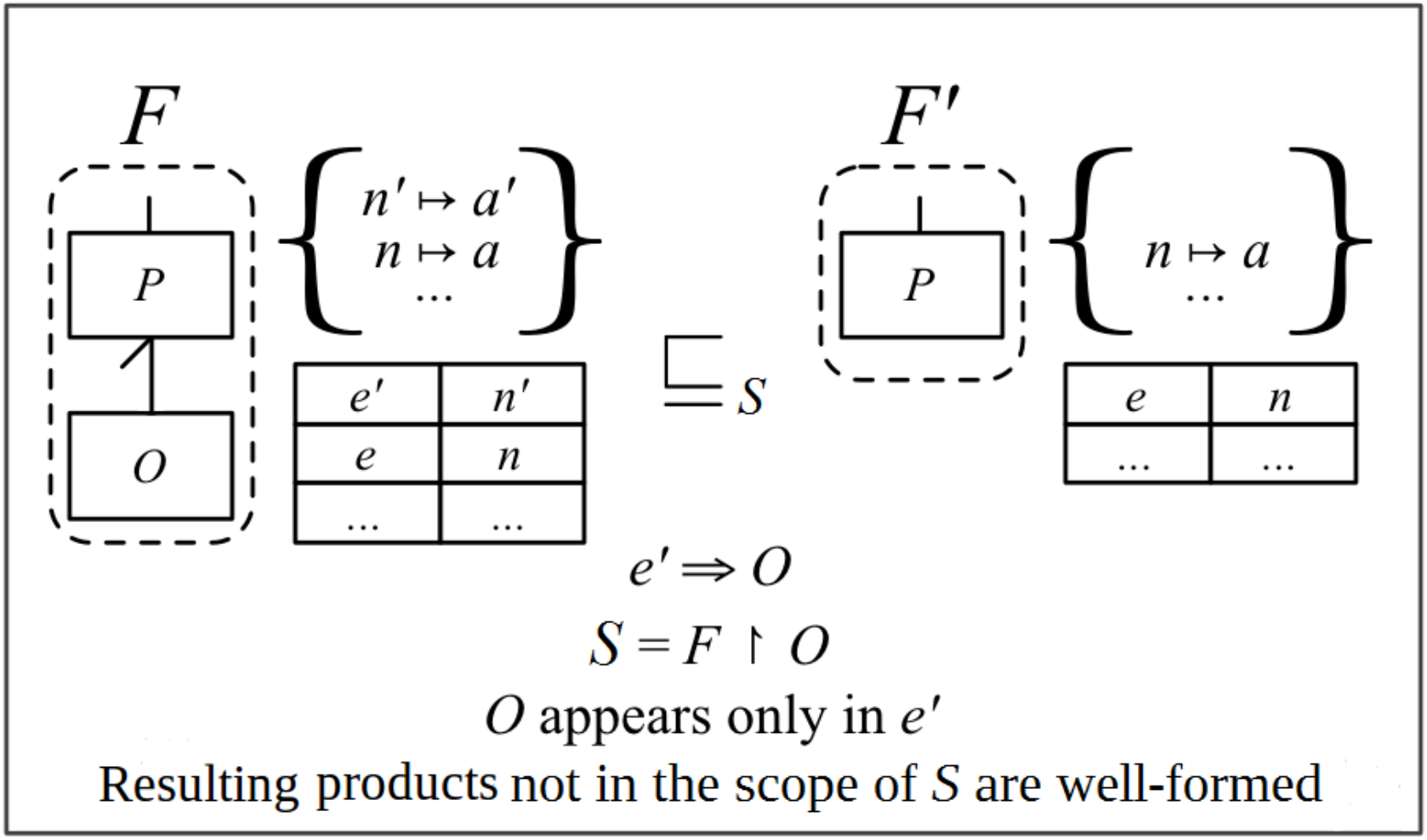
14 include/linux/platform_data/leds-renesas-tpu.h

Commit ae3e4c2776

Alternative Notation



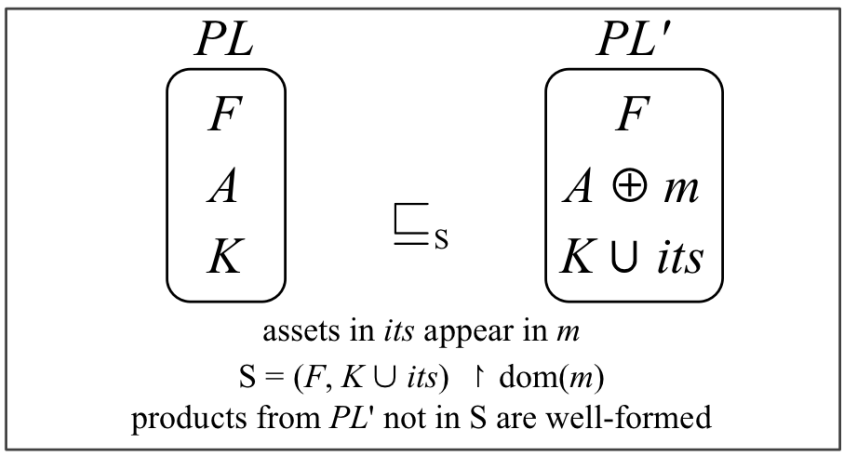
Remove Feature Template



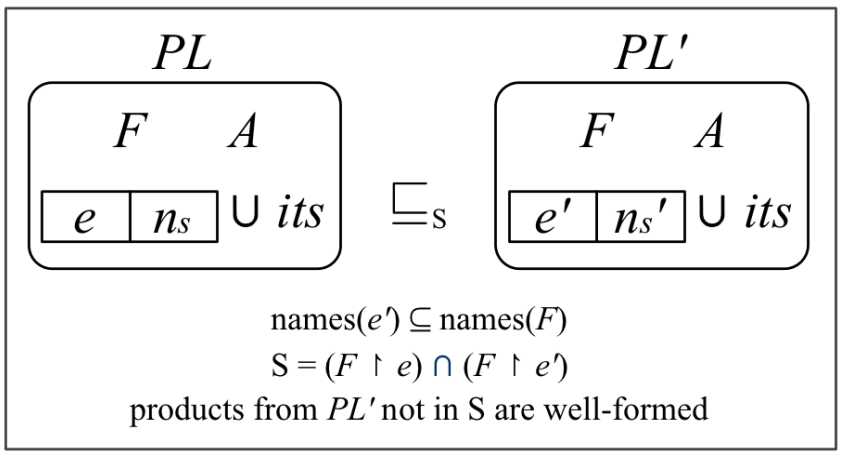
$F \upharpoonright O$ is the set of configurations that are generated from F and do not have O

Other Compositional Templates

Add/Remove Assets Template

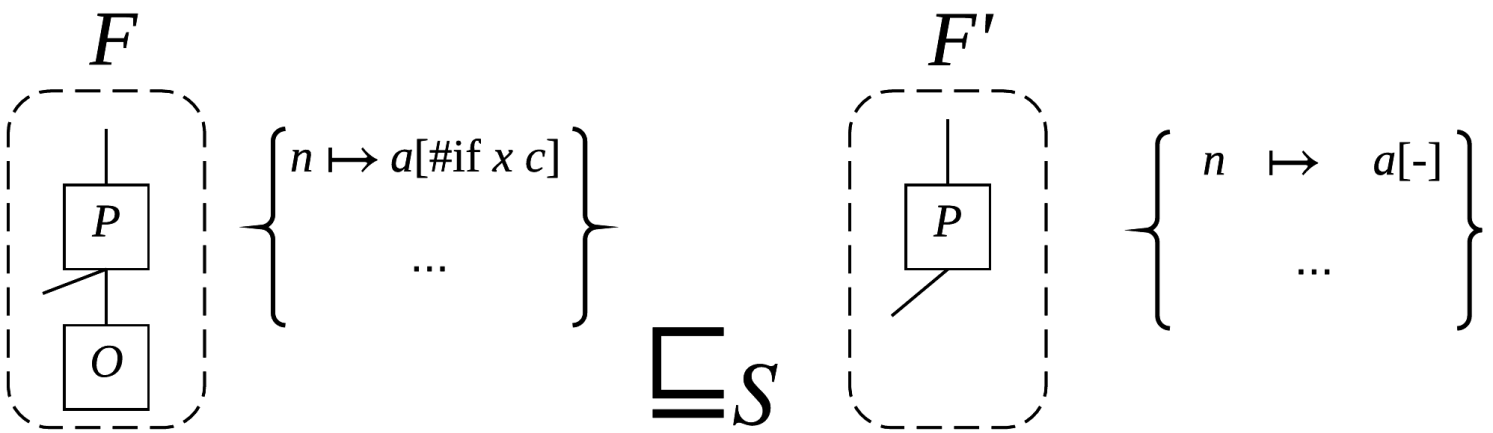


Change/Add/Remove CK Lines Template



Transformational Templates

Remove Feature Transformational Template



e	tag x
e	preprocess n
...	...

...	...

$e \Rightarrow O$

$S = F \restriction O$

O appears only in e

x does not appear in other CK lines

Resulting products not in the scope of S are well-formed

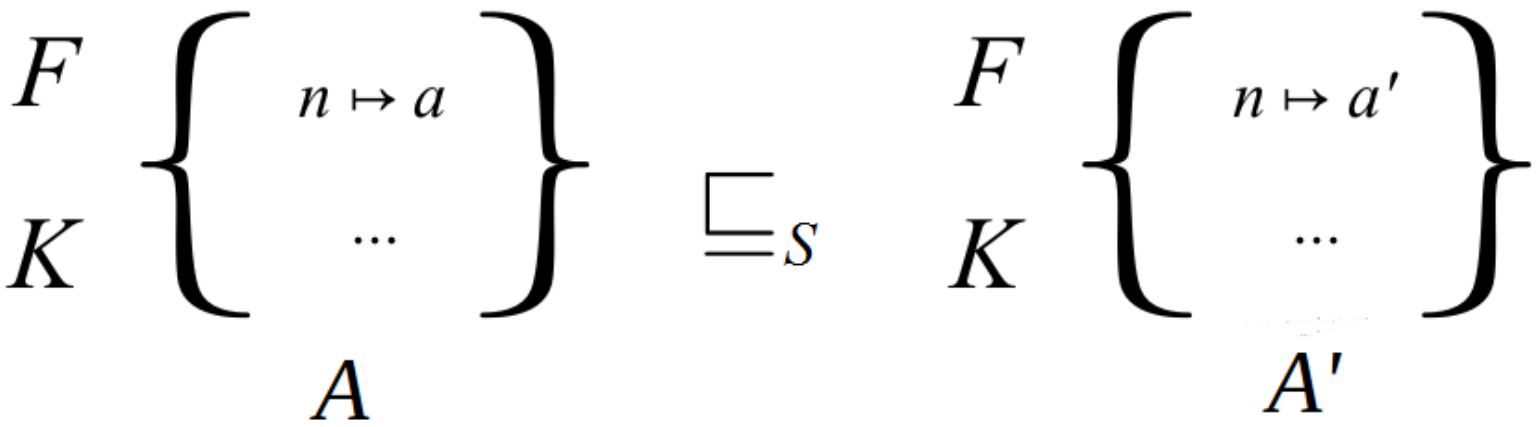
Change Asset Example

1  net/netfilter/ipvs/ip_vs_conn.c

```
if  (cp -> flags & IP_VS_CONN_F_NFCT)  {
-           ip_vs_conn_drop_conntrack(cp);

...
}
```

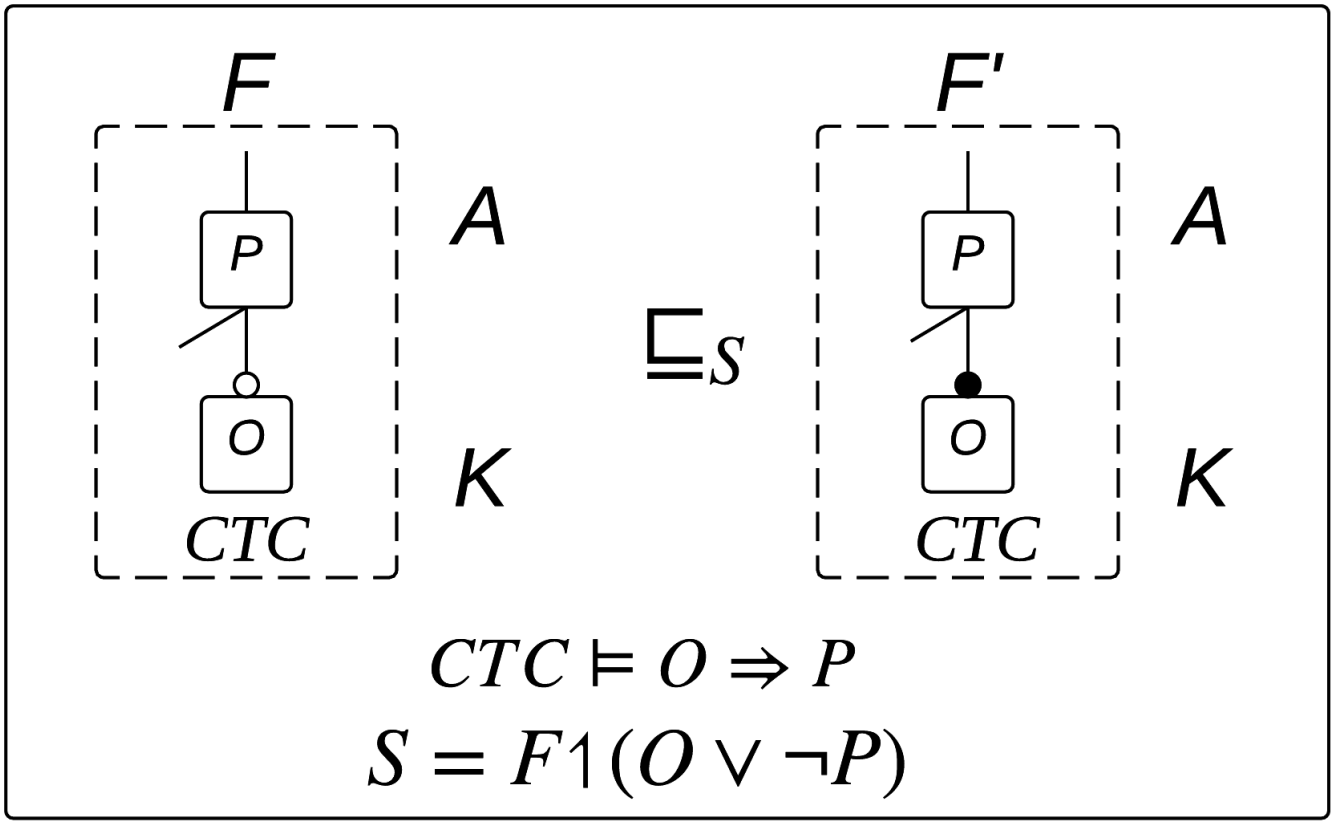
Commit 2627b7e15c

Change Asset Template



$(F, A, K) \upharpoonright \{n\}$ is the set of configurations that are generated from F whose features are not implemented by n

Transform Optional to Mandatory Template



Evaluation

Main Goal

Discover whether the proposed templates would be applicable in a product line development context

Research Question

How often are partially safe evolution templates applicable in a software product line project?

Metric

Number of partially safe evolution scenarios that match each template

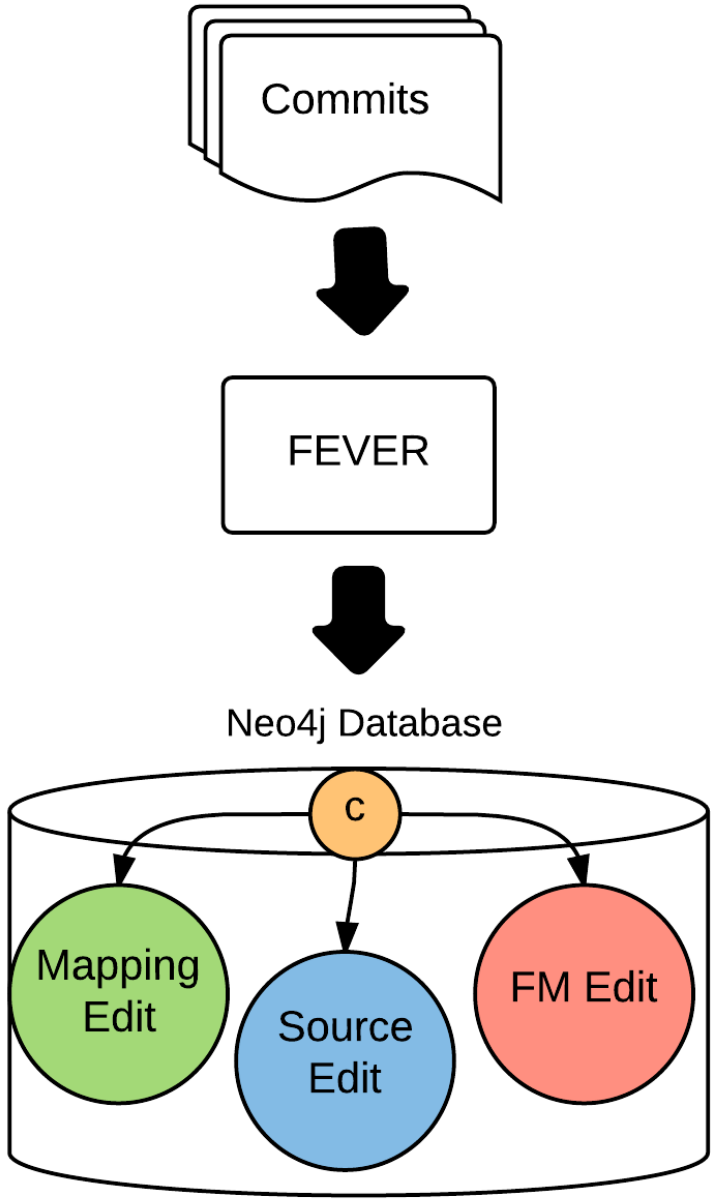
Projects Analysed

15.373 KLOC
43036 Stars
16.323 Contributors
67310 Commits analysed
2 Sep 2013 - 3 Aug 2014
Versions 3.11 - 3.16

170 KLOC
149 Stars
47 Contributors
2300 Commits analysed
26 Jun 2015 - 9 Apr 2016
Versions v1_beta0 - v1_beta18

http://github.com/torvalds/linux
http://github.com/solettaproject/soletta

FEVER Tool



[Dintzner et al. MSR 2016] FEVER: Extracting Feature-oriented Changes from Commits

Templates Identified

- Change Asset
- Add Assets
- Remove Assets
- Change CK Line
- Add CK Lines
- Remove CK Lines
- Remove Feature
- ~~Transform optional to mandatory~~
- ~~Move feature~~

Query Example

Change Asset Template

```
MATCH (c:commit)-[:TOUCHES]->(file:ArtefactEdit)
WHERE
  NOT (c)-[:CHANGES_VM]->(:FeatureEdit) AND
  NOT (c)-[:CHANGES_BUILD]->(:MappingEdit) AND
  file.type="source" AND
  NOT (c)-[:TOUCHES]->(:ArtefactEdit {type:"source"
      ,change:"ADDED"}) AND
  NOT (c)-[:TOUCHES]->(:ArtefactEdit {type:"source"
      ,change:"REMOVED"})
RETURN DISTINCT c
```

Change Asset Example

1  net/netfilter/ipvs/ip_vs_conn.c

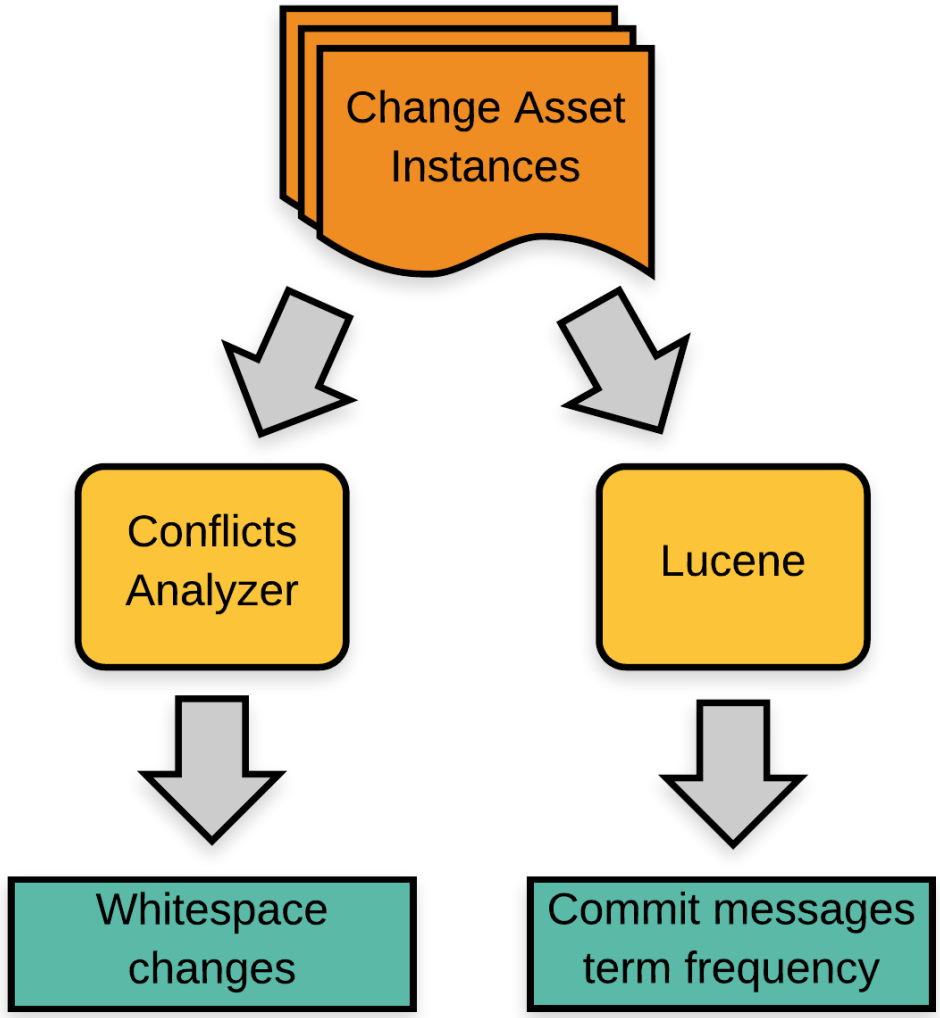
```
if  (cp -> flags & IP_VS_CONN_F_NFCT)  {
-           ip_vs_conn_drop_conntrack(cp);

...
}
```

Commit 2627b7e15c

Results - Linux

Template	Found	Excl.	Problems	Remaining
Change Asset	55345	780	Change type	54565 (89.4%)
Add Assets	181	13	Dataset	168 (0.27%)
Remove Assets	17	1	Dataset	16 (0.02%)
Change CK Line	18	0	Query	18 (0.02%)
Add CK Lines	9	0	Query	9 (0.01%)
Remove CK Lines	12	0	Query	12 (0.02%)
Remove Feature	93	25	Query	68 (0.11%)

Reducing Change Asset Imprecision



http://lucene.apache.org/core/

http://github.com/prga/conflictsAnalyzer

Term Frequency Analysis

Term	Frequency	Rank
Use	12609	1
Fix	11836	2
Patch	9921	3
Add	9916	4
Remove	8352	8
Error	4200	41
Change	4131	42
Bug	1870	146
Failure	1228	267
Rename	1111	305
Modify	431	954
Refactor	422	976

Linux Results Summary

Merge Commits	5413
Analysed Commits	61897
Total	67310

Supported	Commits
Yes	55635 (89,88%)
No	6262 (10,12%)

The 10% include, among others:

- Safe evolution scenarios
- Changes only to the FM
- Changes only to non-source files

Results - Soletta

Template	Found	Excl.	Problems	Remaining
Change Asset	1496	0	Change type	1496 (65%)
Add Assets	5	0	Dataset	5 (0.22%)
Remove Assets	0	0	Dataset	0 (0%)
Change CK Line	9	0	Query	9 (0.39%)
Add CK Lines	3	0	Query	3 (0.13%)
Remove CK Lines	0	0	Query	0 (0%)
Remove Feature	5	3	Query	2 (0.09%)

Soletta Results Summary

Merge Commits	1
Analysed Commits	2299
Total	2300

Supported	Commits
Yes	1515 (65,89%)
No	785 (34,14%)

The 34% include, among others:

- Safe evolution scenarios
- Changes only to the FM
- Changes only to non-source files

Threats to Validity

Construct

Some scenarios actually represent safe evolution
Eg.: asset refinement

Internal

Possible bugs in FEVER
Queries precision
Manual analysis

External

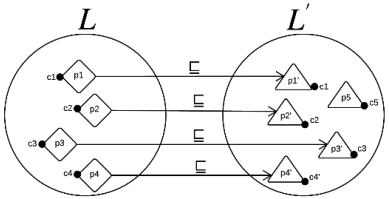
Other projects
Other periods of the Linux project

Conclusion

SPL Refinement

$L \sqsubseteq L'$ whenever

$\forall c \in [\![F(L)]\!] \cdot \exists c' \in [\![F(L')]\!] \cdot prod(L, c) \sqsubseteq prod(L', c')$



Feature Removal Scenario

12 drivers/leds/Kconfig

```
- config LEDS_RENESAS_TPU
-       bool "LED support for Renesas TPU"
-       depends on LEDS_CLASS=y && HAVE_CLK && GPIOLIB
-       help
-         ...
```

1 drivers/leds/Makefile

```
- obj-$(CONFIG_LEDS_RENESAS_TPU) += leds-renesas-tpu.o
```

337 drivers/leds/leds-renesas-tpu.c

14 include/linux/platform_data/leds-renesas-tpu.h

Commit ae3e4c2776

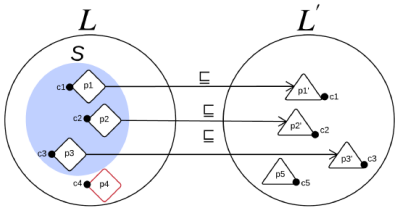
Partial Refinement

$L \sqsubseteq_S L'$ whenever

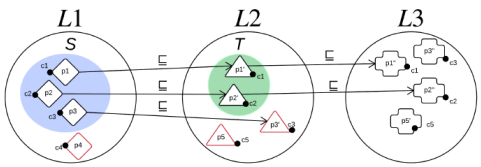
$S \subseteq [\![F(L)]\!] \wedge S \subseteq [\![F(L')]\!]$

and

$\forall c \in S \cdot prod(L, c) \sqsubseteq prod(L', c)$



Partial Refinement is a Pre-order

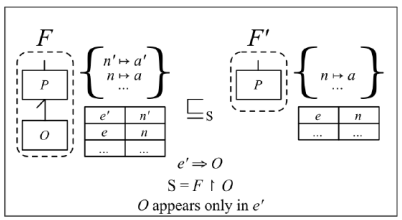


$L1 \sqsubseteq_S L2 \wedge L2 \sqsubseteq_T L3$

$\Rightarrow$

$L1 \sqsubseteq_{S \cap T} L3$

Remove Feature Template



$F \upharpoonright O$ is the set of configurations that are generated from F and do not have O

Evaluation

Main Goal

Discover whether the proposed templates are frequently applicable in a product line development context

Research Question

How often are partially safe evolution templates applicable in a software product line project?

Metric

Number of partially safe evolution scenarios that match each template

Future Work

- Derive other properties
- Define other templates
- Deeper evaluation
- Tool to support developers

Partially Safe Evolution of SPLs

Gabriela Cunha Sampaio

Supervisor: Paulo Henrique Monteiro Borba

Co-supervisor: Leopoldo Motta Teixeira

March 27, 2017