The Variability Model of the Linux Kernel

Steven She, Rafael Lotufo, Thorsten Berger,
Krzysztof Czarnecki, Andrzej Wąsowski

University of Waterloo
University of Leipzig
IT University of Copenhagen

January 26, 2010

Motivation

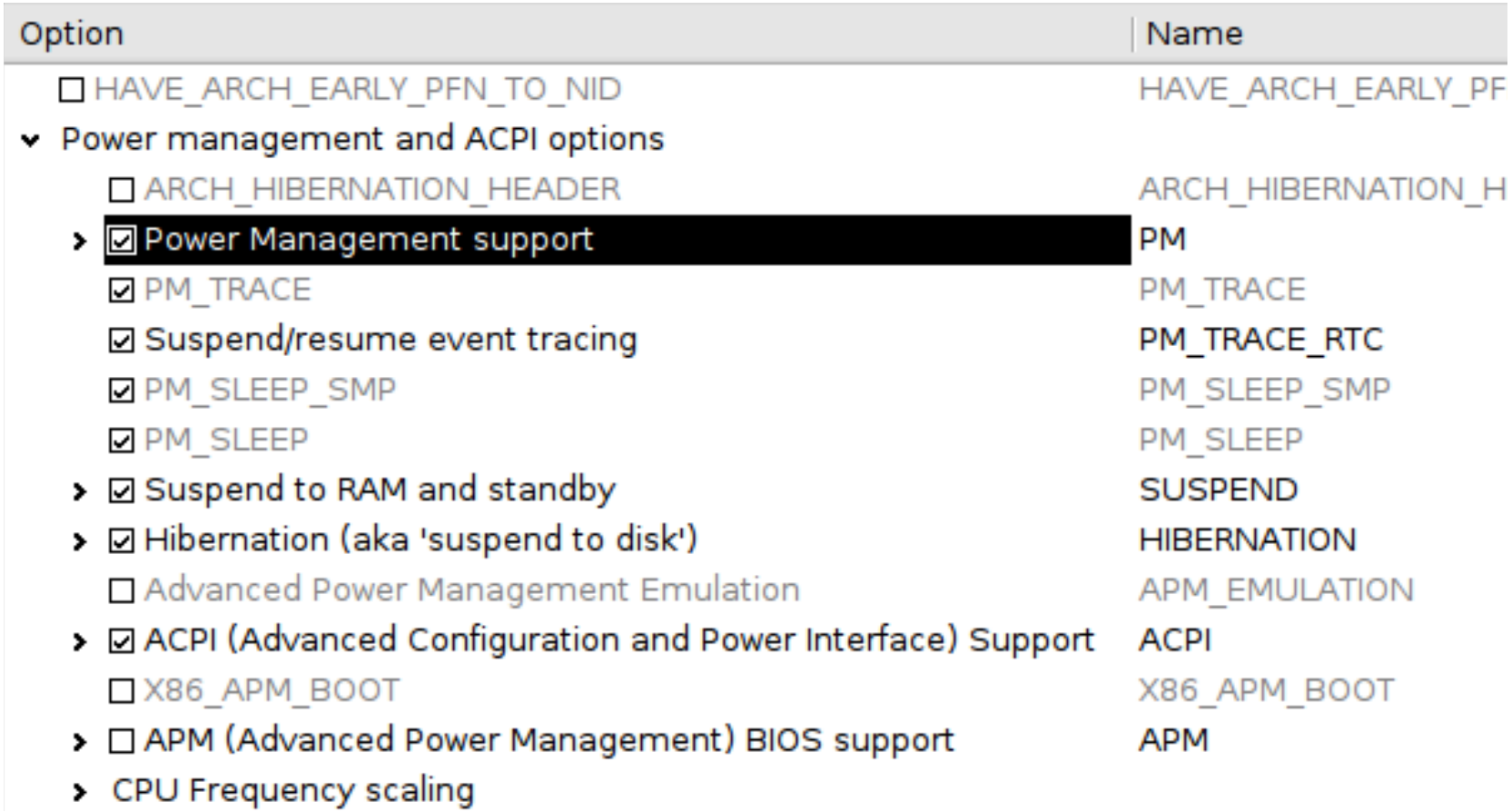
There is a growing number of variability modeling tools.
We need **practical benchmarks** to evaluate these tools.

Motivation

There is a growing number of variability modeling tools.
We need **practical benchmarks** to evaluate these tools.

Linux is a great example of a real software product line
with a explicit variability model!

Linux configurator



Variability model is specified using the **Kconfig** language.

Kconfig declaration

```
menu "Power management and ACPI options"
  depends on !X86_VOYAGER

  config PM
    bool "Power Management support"
    depends on !IA64_HP_SIM
    ---help---
        "Power Management" means that …

  config PM_DEBUG
    bool "Power Management Debug Support"
    depends on PM

  config CPU_IDLE
    bool "CPU idle PM support"
    default ACPI
endmenu
```

Kconfig declaration

```
menu "Power management and ACPI options"
  depends on !X86_VOYAGER

  config PM
    bool "Power Management support"
    depends on !IA64_HP_SIM
    ---help---
        "Power Management" means that …

  config PM_DEBUG
    bool "Power Management Debug Support"
    depends on PM

  config CPU_IDLE
    bool "CPU idle PM support"
    default ACPI
endmenu
```

Kconfig declaration

```
menu "Power management and ACPI options"
  depends on !X86_VOYAGER

  config PM
    bool "Power Management support"
    depends on !IA64_HP_SIM
    ---help---
        "Power Management" means that …

  config PM_DEBUG
    bool "Power Management Debug Support"
    depends on PM

  config CPU_IDLE
    bool "CPU idle PM support"
    default ACPI
endmenu
```

Kconfig declaration

```
menu "Power management and ACPI options"
  depends on !X86_VOYAGER

  config PM
    bool "Power Management support"
    depends on !IA64_HP_SIM
    ---help---
        "Power Management" means that …

  config PM_DEBUG
    bool "Power Management Debug Support"
    depends on PM

  config CPU_IDLE
    bool "CPU idle PM support"
    default ACPI
endmenu
```

Kconfig declaration

```
menu "Power management and ACPI options"
  depends on !X86_VOYAGER

  config PM
    bool "Power Management support"
    depends on !IA64_HP_SIM
    ---help---
        "Power Management" means that …

  config PM_DEBUG
    bool "Power Management Debug Support"
    depends on PM

  config CPU_IDLE
    bool "CPU idle PM support"
    default ACPI
endmenu
```

Kconfig declaration

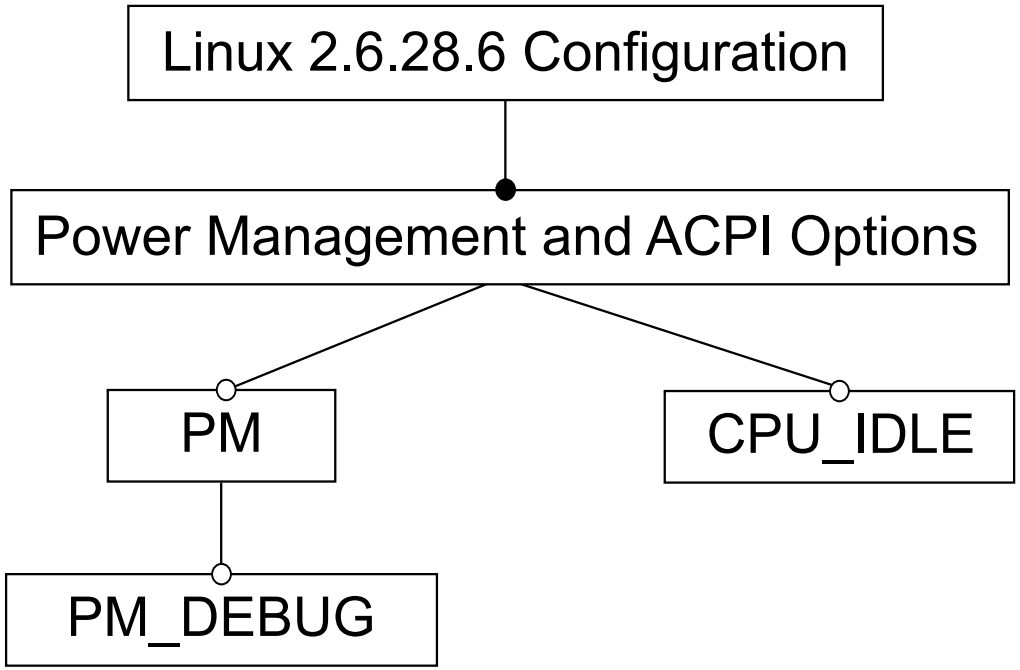
```
menu "Power management and ACPI options"
  depends on !X86_VOYAGER

  config PM
    bool "Power Management support"
    depends on !IA64_HP_SIM
    ---help---
        "Power Management" means that …

  config PM_DEBUG
    bool "Power Management Debug Support"
    depends on PM

  config CPU_IDLE
    bool "CPU idle PM support"
    default ACPI
endmenu
```

Kconfig feature model



$$\text{PM\_MENU} \rightarrow \neg\text{X86\_VOYAGER}$$
$$\text{PM} \rightarrow \neg\text{IA64\_HP\_SIM}$$
$$\text{ACPI} \leftrightarrow \text{CPU\_IDLE}$$

Linux KConfig → Feature Model

Analyzed four aspects of the Linux 2.6.28.6 Kconfig model in terms of feature modeling concepts:

- characterized features,
- model hierarchy,
- constraints,
- and natural language properties.

Comparing with published models

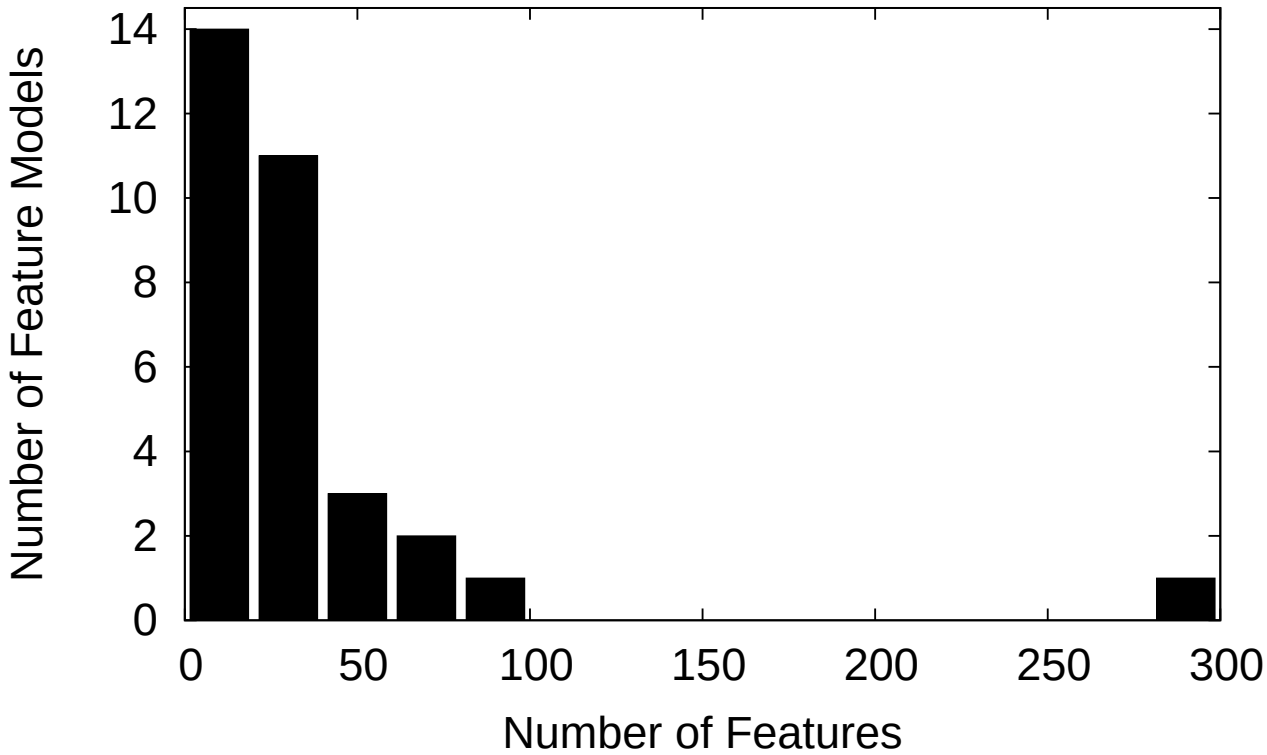
Compared Linux statistics with 32 published models[1].

- 19 models - software product lines
- 8 models - other product lines (e.g. hardware, business)
- 5 models - domain models (e.g. eCommerce systems)

Only 5 models describe real, existing software systems.

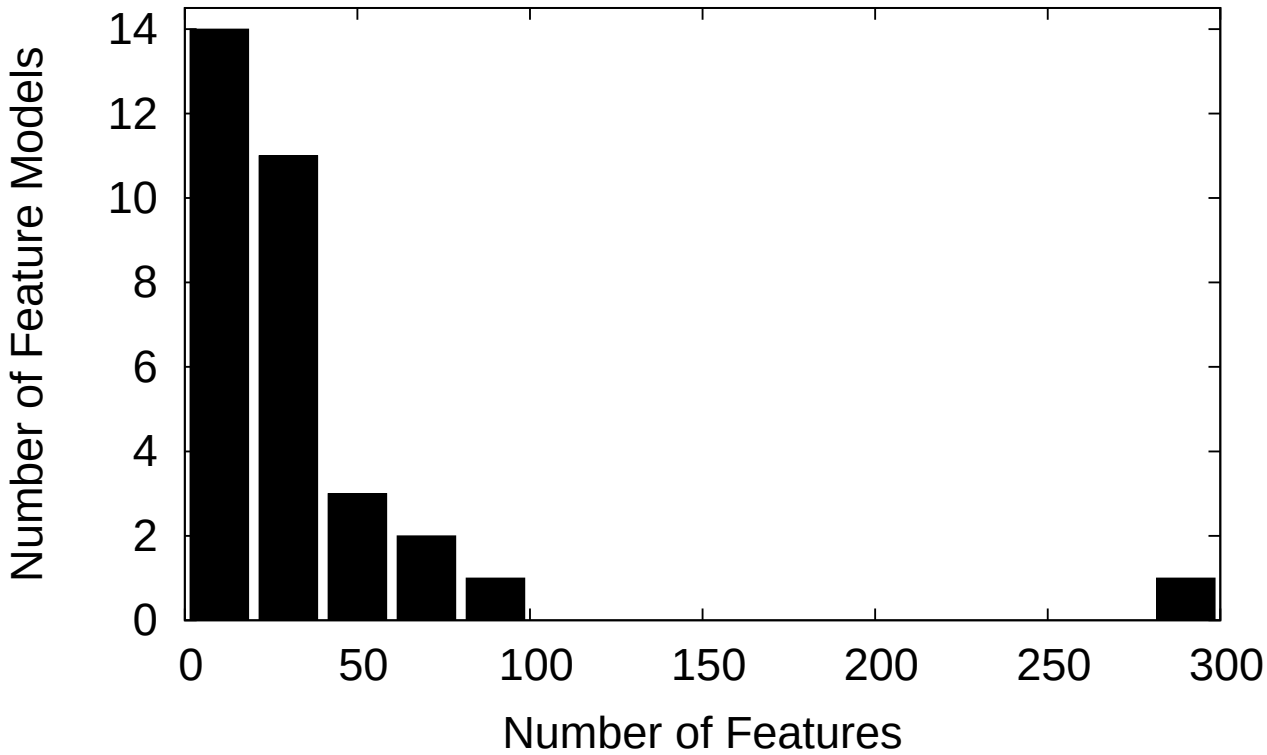
[1] http://www.splot-research.org

Size of published models



Linux Kconfig model has 5426 features.

Size of published models



Linux Kconfig model has 5426 features.

Linux feature statistics

Kconfig Concept	Features	Mand.	Grouped	XOR + OR
Config	5323	0	146	0
Non / User-Sel.	547 + 4744			
Boolean	2005	0	136	0
Tristate	3130	0	10	0
Int	132	132	0	0
Hex	29	29	0	0
String	27	27	0	0
Menu	71	38	0	0
Choice	32	31	0	30 + 2
Total	**5426**	257	146	30 + 2

Linux feature statistics

Kconfig Concept	Features	Mand.	Grouped	XOR + OR
Config	5323	0	146	0
Non / User-Sel.	547 + 4744			
Boolean	2005	0	136	0
Tristate	3130	0	10	0
Int	132	132	0	0
Hex	29	29	0	0
String	27	27	0	0
Menu	71	38	0	0
Choice	32	31	0	30 + 2
Total	**5426**	257	146	30 + 2

Linux feature statistics

Kconfig Concept	Features	Mand.	Grouped	XOR + OR
Config	5323	0	146	0
Non / User-Sel.	547 + 4744			
Boolean	2005	0	136	0
Tristate	3130	0	10	0
Int	132	132	0	0
Hex	29	29	0	0
String	27	27	0	0
Menu	71	38	0	0
Choice	32	31	0	30 + 2
Total	**5426**	257	146	30 + 2

Linux feature statistics

Kconfig Concept	Features	Mand.	Grouped	XOR + OR
Config	5323	0	146	0
Non / User-Sel.	547 + 4744			
Boolean	**2005**	0	**136**	0
Tristate	**3130**	0	**10**	0
Int	132	132	0	0
Hex	29	29	0	0
String	27	27	0	0
Menu	71	38	0	0
Choice	32	31	0	30 + 2
Total	**5426**	257	146	30 + 2

Linux feature statistics

Kconfig Concept	Features	Mand.	Grouped	XOR + OR
Config	5323	0	146	0
Non / User-Sel.	547 + 4744			
Boolean	2005	0	136	0
Tristate	3130	0	10	0
Int	**132**	**132**	0	0
Hex	**29**	**29**	0	0
String	**27**	**27**	0	0
Menu	71	38	0	0
Choice	32	31	0	30 + 2
Total	**5426**	257	146	30 + 2

Linux feature statistics

Kconfig Concept	Features	Mand.	Grouped	XOR + OR
Config	5323	0	146	0
Non / User-Sel.	547 + 4744			
Boolean	2005	0	136	0
Tristate	3130	0	10	0
Int	132	132	0	0
Hex	29	29	0	0
String	27	27	0	0
Menu	71	38	0	0
Choice	32	31	0	30 + 2
Total	**5426**	257	146	30 + 2

Linux feature statistics

Kconfig Concept	Features	Mand.	Grouped	XOR + OR
Config	5323	0	146	0
Non / User-Sel.	547 + 4744			
Boolean	2005	0	136	0
Tristate	3130	0	10	0
Int	132	132	0	0
Hex	29	29	0	0
String	27	27	0	0
Menu	71	38	0	0
Choice	32	31	0	30 + 2
Total	**5426**	257	146	30 + 2

Linux feature statistics

Kconfig Concept	Features	Mand.	Grouped	XOR + OR
Config	5323	0	146	0
Non / User-Sel.	547 + 4744			
Boolean	2005	0	136	0
Tristate	3130	0	10	0
Int	132	132	0	0
Hex	29	29	0	0
String	27	27	0	0
Menu	71	38	0	0
Choice	32	31	0	30 + 2
Total	**5426**	257	146	30 + 2

Published models vs. Linux

Concept	Published Models (%)			Linux (%)
	median	min	max	
mandatory features	25	0	66	4.74
grouped features	44	0	75	2.69
groups	16	0	35	0.59
XOR	9	0	30	0.55
OR	6	0	16	0.04

Code-granularity of features

Two heuristics for automatic feature selection in the Linux configurator: *allyes*, *allno*.

Metric	allyes	allno	Δ	θ
Features	3,448	61	3387	1
Files	10,326	973	9,353	2.76
SLOC	4,266,171	210,302	4,055,869	1,197.48

$$\Delta_i = \textbf{allyes}_i - \textbf{allno}_i; \theta_i = \Delta_i / \Delta_1$$

Code-granularity of features

Two heuristics for automatic feature selection in the Linux configurator: *allyes*, *allno*.

Metric	**allyes**	**allno**	Δ	θ
Features	3,448	61	3387	1
Files	10,326	973	9,353	2.76
SLOC	4,266,171	210,302	4,055,869	1,197.48

$$\Delta_i = \textbf{allyes}_i - \textbf{allno}_i; \theta_i = \Delta_i / \Delta_1$$

Qualitative characteristics

Subjective categorization of 180 randomly selected features.

Menu	Support	Option	Debug	No Info
1	97	46	13	23

User-based granularity categories of Linux features

API	Driver	Kernel	Protocol	Subsystem	No Info
5	120	15	14	1	25

Types of features in Linux kernel

Qualitative characteristics

Subjective categorization of 180 randomly selected features.

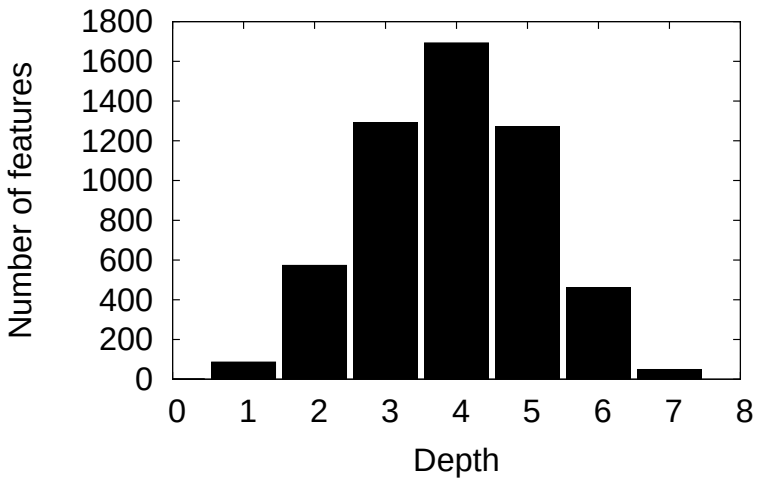
Menu	Support	Option	Debug	No Info
1	97	46	13	23

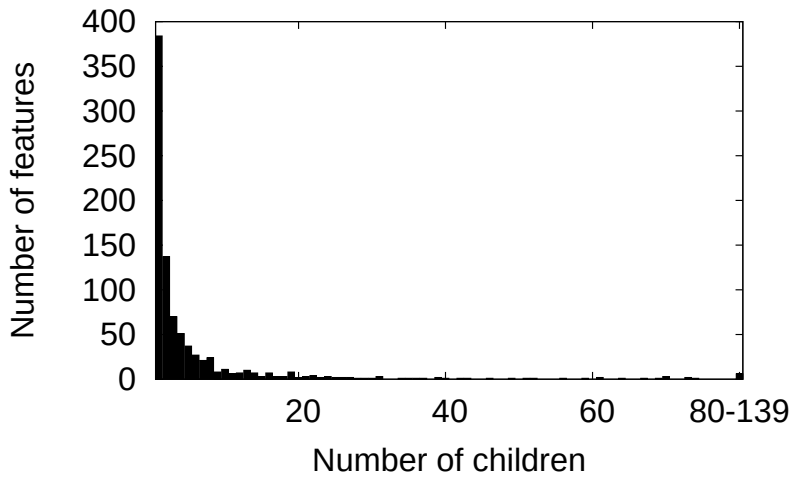
User-based granularity categories of Linux features

API	Driver	Kernel	Protocol	Subsystem	No Info
5	120	15	14	1	25

Types of features in Linux kernel

Hierarchy statistics

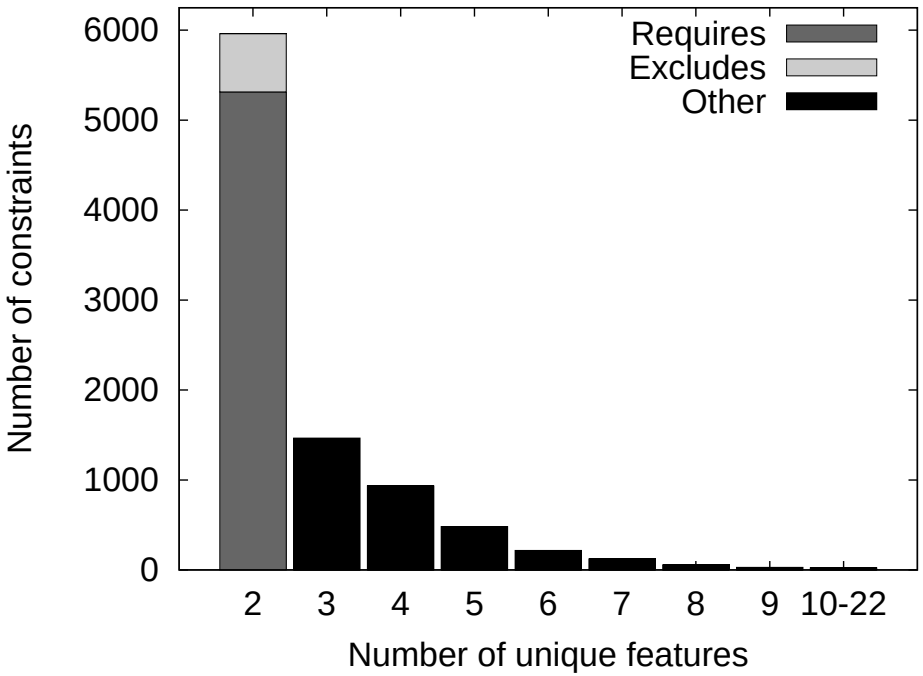




- 4544 leaves
- well-balanced
- relatively shallow

- many single childs
- long tail

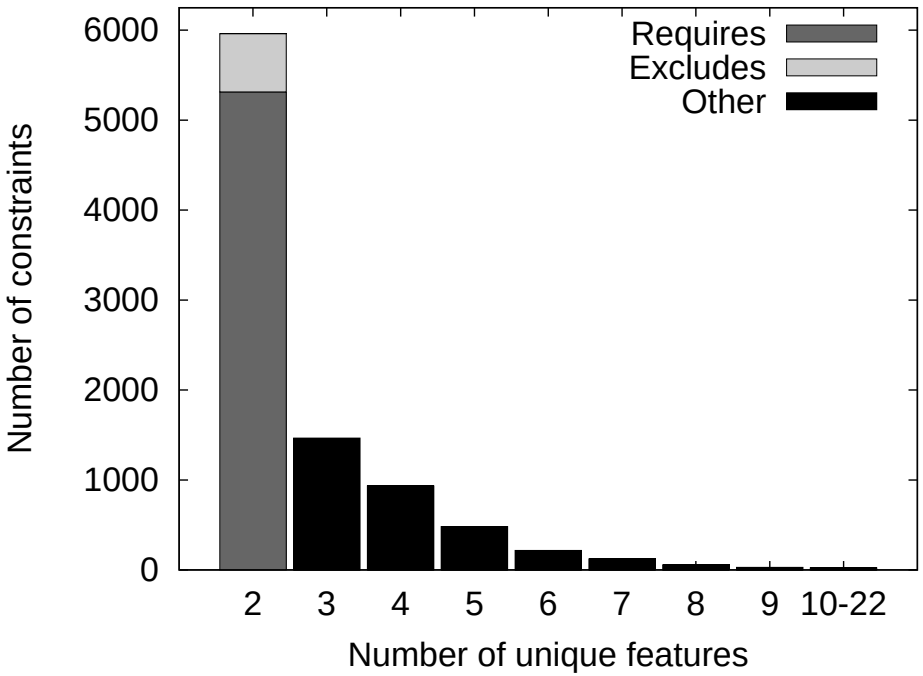
Constraint statistics



- 9291 constraints
- 82% features referenced
- 89% requires constraints
- some v. large constraints

Significant number of cross-tree constraints!

Constraint statistics



- 9291 constraints
- 82% features referenced
- 89% requires constraints
- some v. large constraints

Significant number of cross-tree constraints!

Identifiers, prompts and descriptions

```
config PM
  bool "Power Management support"
  depends on !IA64_HP_SIM
  ---help---
```

"Power Management" means that parts of your computer are shut off or put into a power conserving "sleep" mode if they are not being used. There are two competing standards for doing this: APM and ACPI. If you want to use either one, say Y here and then also to the requisite support below.

Power Management is most important for battery powered laptop computers; if you have a laptop, check out the Linux Laptop home page on the WWW at <http://www.linux-on-laptops.com/> or Tuxmobil - Linux on Mobile Computers at <http://www.tuxmobil.org/> and the Battery Powered Linux mini-HOWTO, available from <http://www.tldp.org/docs.htmlhowto>.

Natural language properties

artifact	no. of characters			no. of words		
	median	min	max	median	min	max
identifiers	13	2	58	2	1	9
prompts	27	2	82	4	1	13
description	-	-	-	29	2	392

Size of textual artifacts

text source	most frequent domain terms					
identifier	*usb*	*snd*	*md*	*serial*	*fb*	*debug*
prompt	*usb*	*ethernet*	*pci*	*intel*	*scsi*	*pcmcia*
description	*usb*	*linux*	*scsi*	*ethernet*	*pci*	*howto*

Top domain terms

Natural language properties

artifact	no. of characters			no. of words		
	median	min	max	median	min	max
identifiers	13	2	58	2	1	9
prompts	27	2	82	4	1	13
description	-	-	-	29	2	392

Size of textual artifacts

text source	most frequent domain terms					
identifier	*usb*	*snd*	*md*	*serial*	*fb*	*debug*
prompt	*usb*	*ethernet*	*pci*	*intel*	*scsi*	*pcmcia*
description	*usb*	*linux*	*scsi*	*ethernet*	*pci*	*howto*

Top domain terms

Related work

- Sincero and Schröder-Preikschat (ASPL 2008)
- Tartler et al. - dead feature detection in Linux (FOSD 2009)
- Segura and Ruiz-Cortés - FM benchmarks (VaMoS 2009)

Conclusions

- Low number of mandatory features and groups.
- Each feature crosses roughly 2.8 source files and 1200 SLOC.
- Average leaf depth of 4, many single childs and long tail.
- Significant number and size of cross-tree constraints.

[1] Available for download at `http://fm.gsdlab.org`

Counting Constraints

Counted the number of distinct literals involved in the declaration of a constraint.

$$(MWINCHIP3D \vee gcMCRUSOE \vee gMEFFICEON \vee MCYRIXIII \vee MK7 \vee MK6 \vee MPENTIUM4 \vee MPENTIUMM \vee MPENTIUMIII \vee MPENTIUMII \vee M686 \vee M586MMX \vee M586TSC \vee MK8 \vee MVIAC3\_2 \vee MVIAC7 \vee MGEODEGX1 \vee MGEODE\_LX \vee MCORE2) \wedge \neg X86\_NUMAQ) \vee X86\_64) \rightarrow X86\_TSC = y$$

Figure: Equation with 22 distinct literals

Kconfig declaration (choice)

```
choice
  prompt "Subarchitecture Type"
  default X86_PC

  config X86_PC
   bool "PC-compatible"

  config X86_GENERICARCH
    bool "Generic architecture"
    depends on X86_32

  if X86_GENERICARCH
    config X86_BIGSMP
     bool "Support for big SMP systems with more than 8 CPUs"
     depends on X86_32 && SMP
  endif

endchoice
```

Kconfig → Feature Model mapping

concepts		Feature modeling concepts
Switch config		Optional feature
Entry-field config		Mandatory feature
Conditional menu		Optional feature
Unconditional menu		Mandatory feature
Choice		
Mandatory	⇛	Mandatory feat. + XOR-group
Optional		Optional feat. + XOR-group
Mandatory tristate		Mandatory feat. + OR-group
Optional tristate		Optional feat. + OR-group
Choice config		Grouped feature
Config, menu or choice nesting	⇛	Sub-feature relation
Visibility conditions		
Selects	⇛	Cross-tree constraint
Constraining defaults		