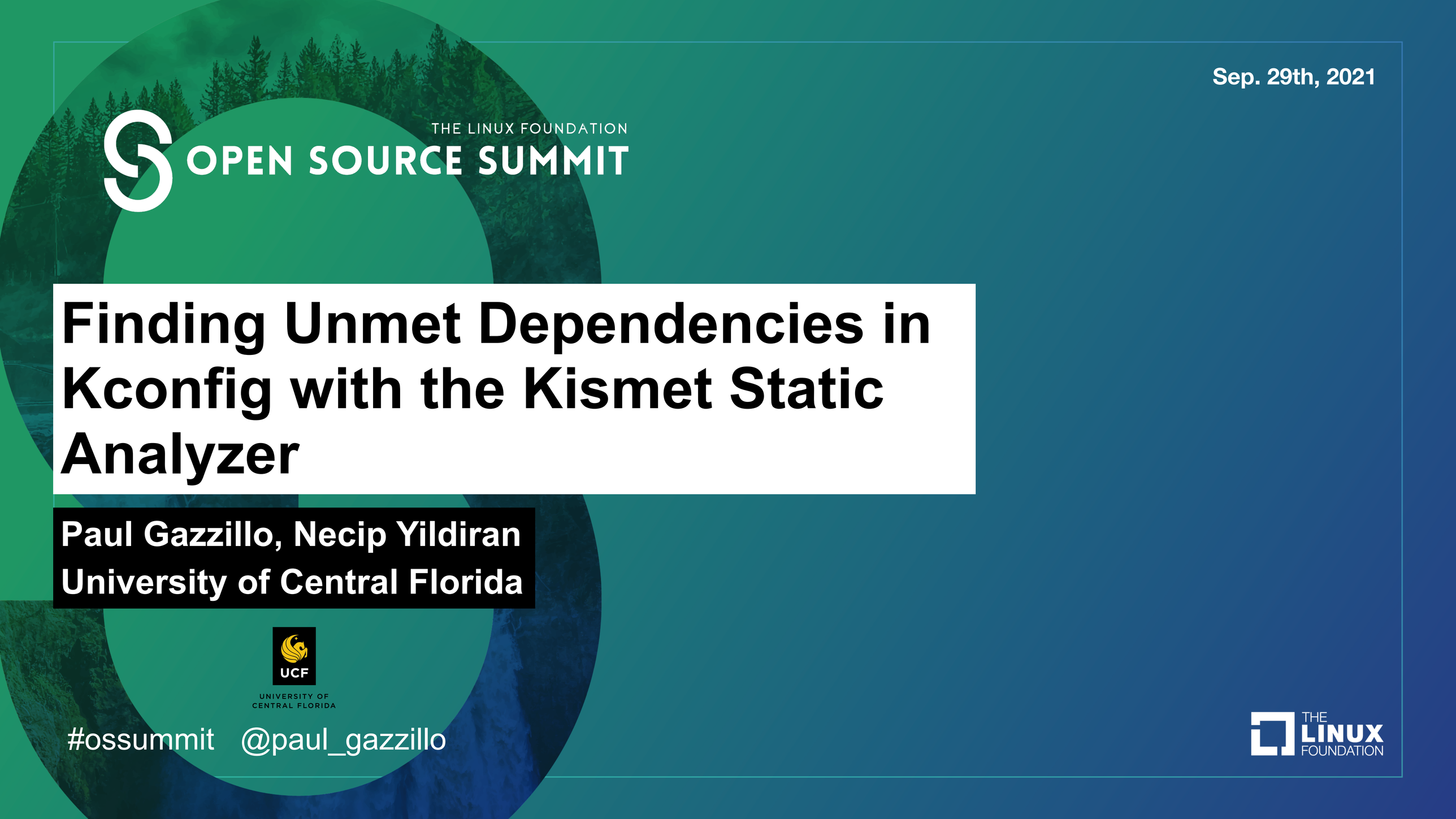


Sep. 29th, 2021

Finding Unmet Dependencies in Kconfig with the Kismet Static Analyzer

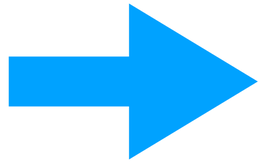
Paul Gazzillo, Necip Yildiran
University of Central Florida

UCF
UNIVERSITY OF CENTRAL FLORIDA

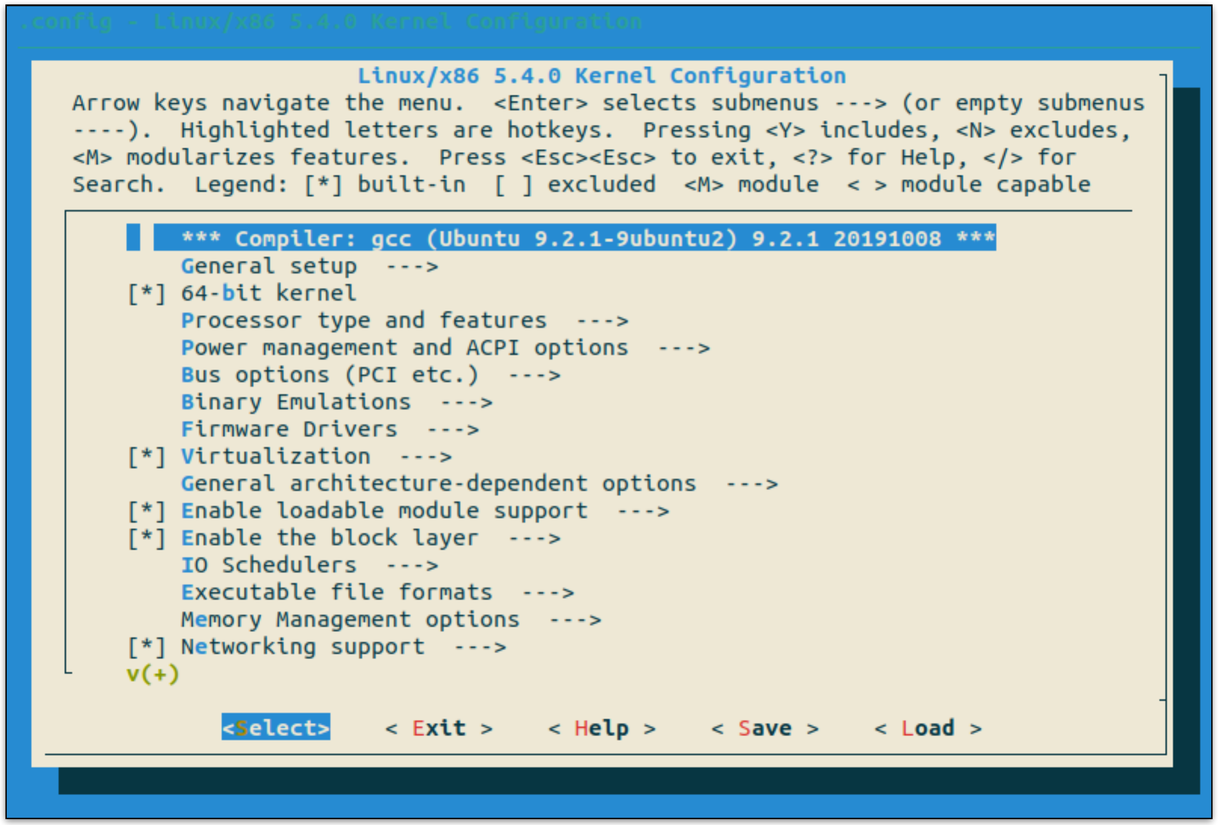
#ossummit @paul_gazzillo

THE LINUX FOUNDATION

linux works on lots of devices



the kernel is ultra-configurable



configurability makes maintenance harder

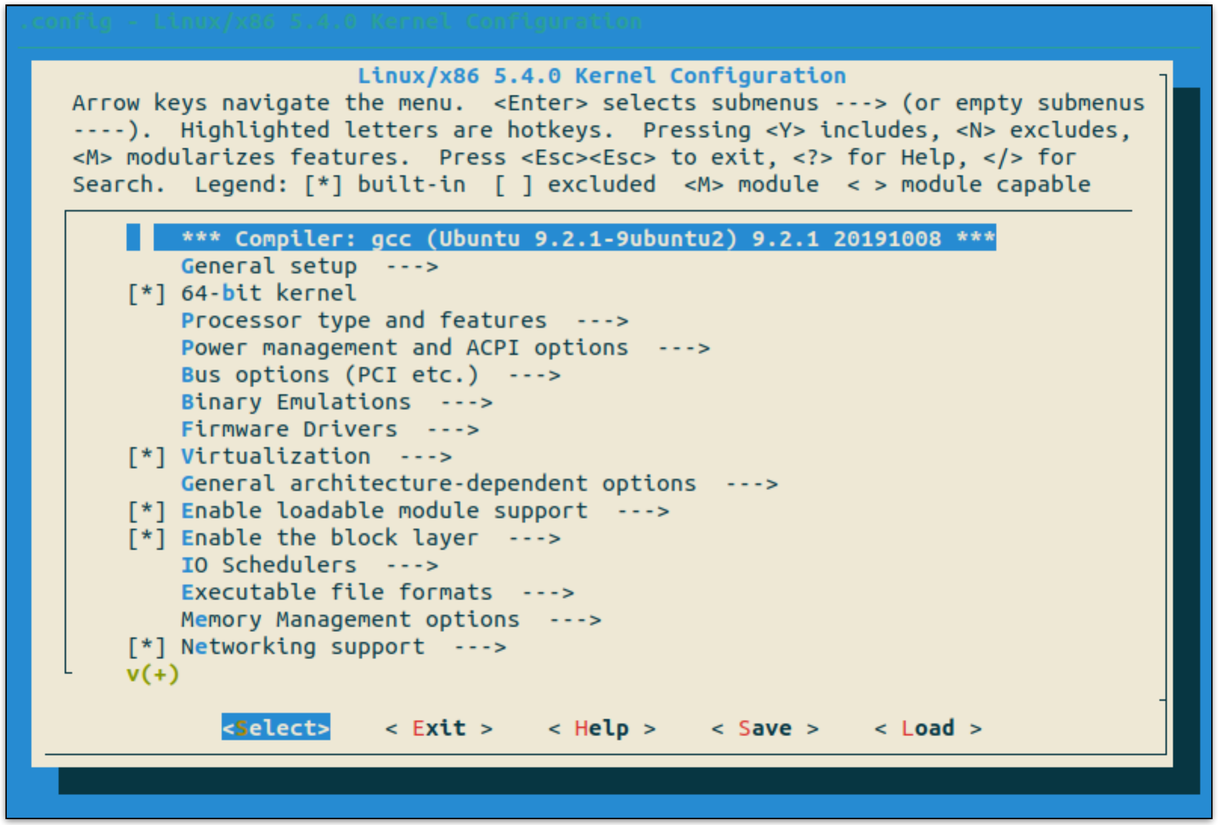
given a patch, what configurations does it affect? (jmake, lawall et al)

given a bug, what configurations does it appear in? (config-bisect)

what's a minimal configuration that includes specific source? (config-bisect)

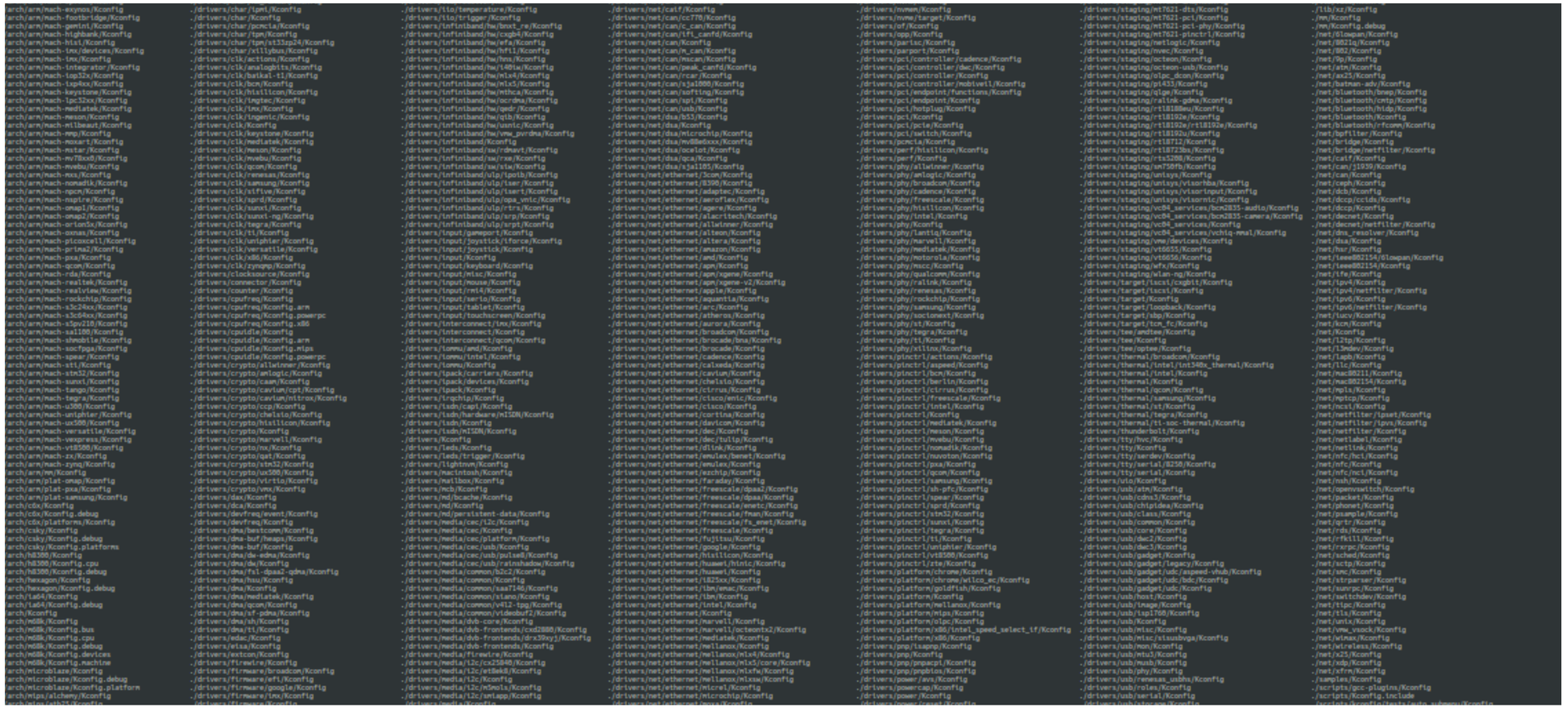
what code is no longer configurable in the kernel? (undertaker, tarlet et al)

there’s about 15,000 configuration options

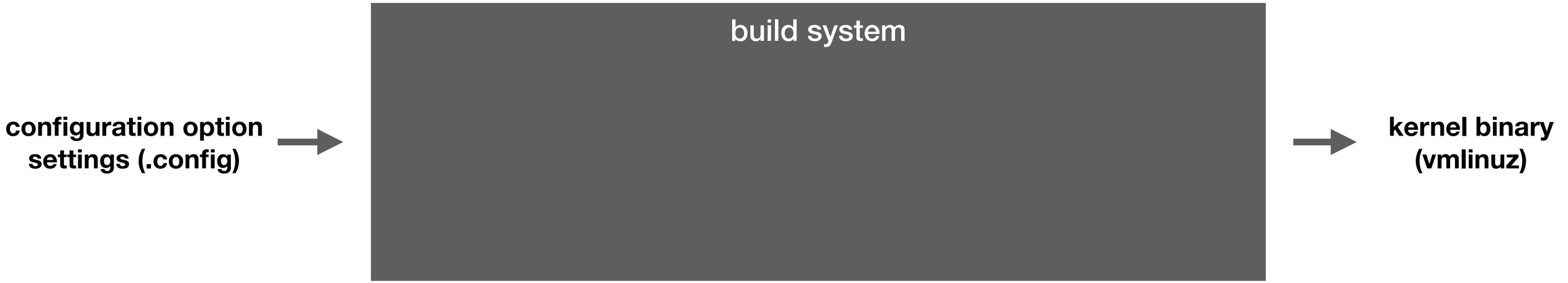


written in about 150,000 lines of Kconfig

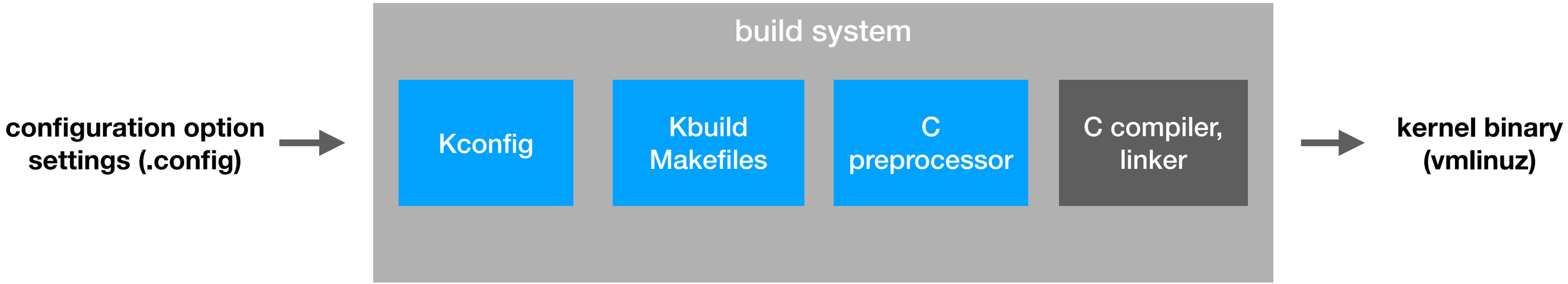
there's around 1,500 Kconfig files



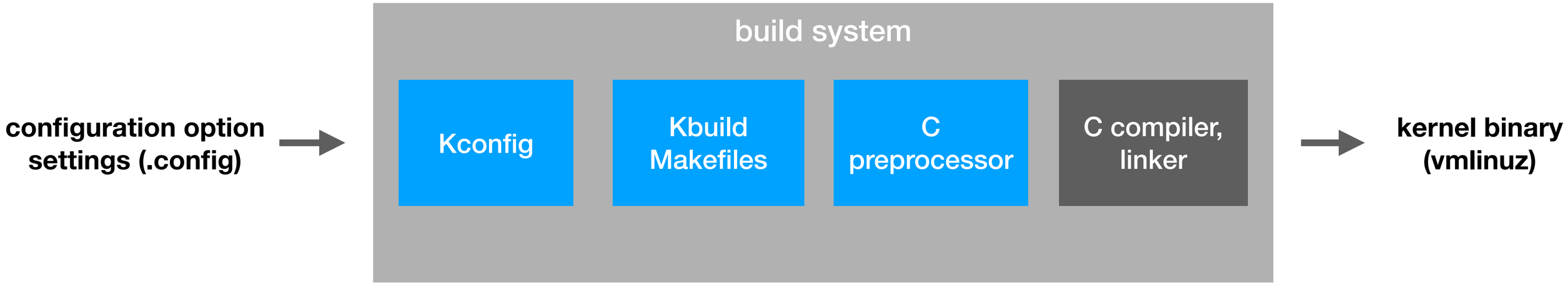
what does the build system do?



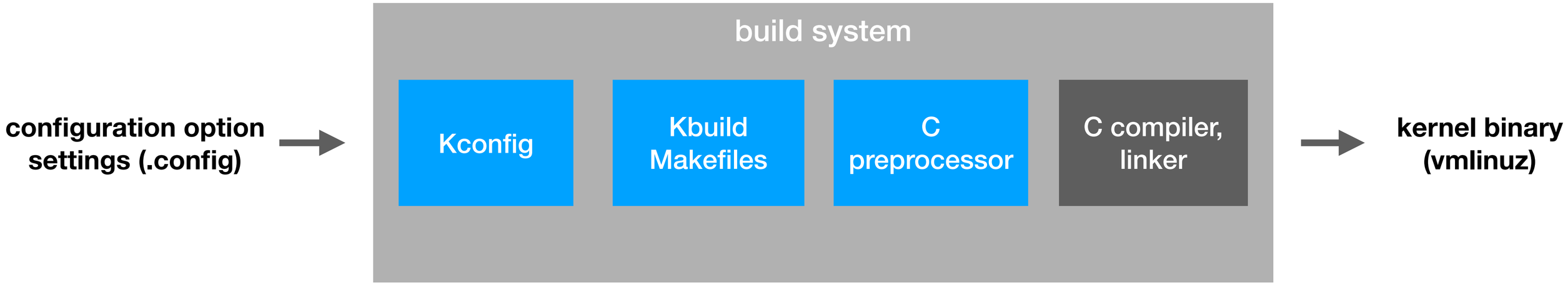
what does the build system do?

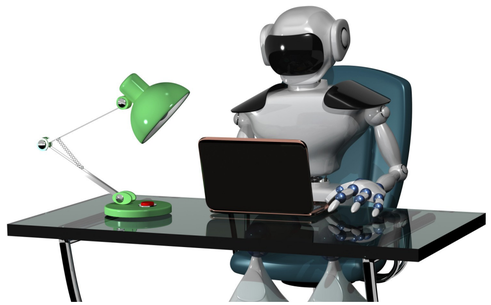


our goal: automatically analyze the build system

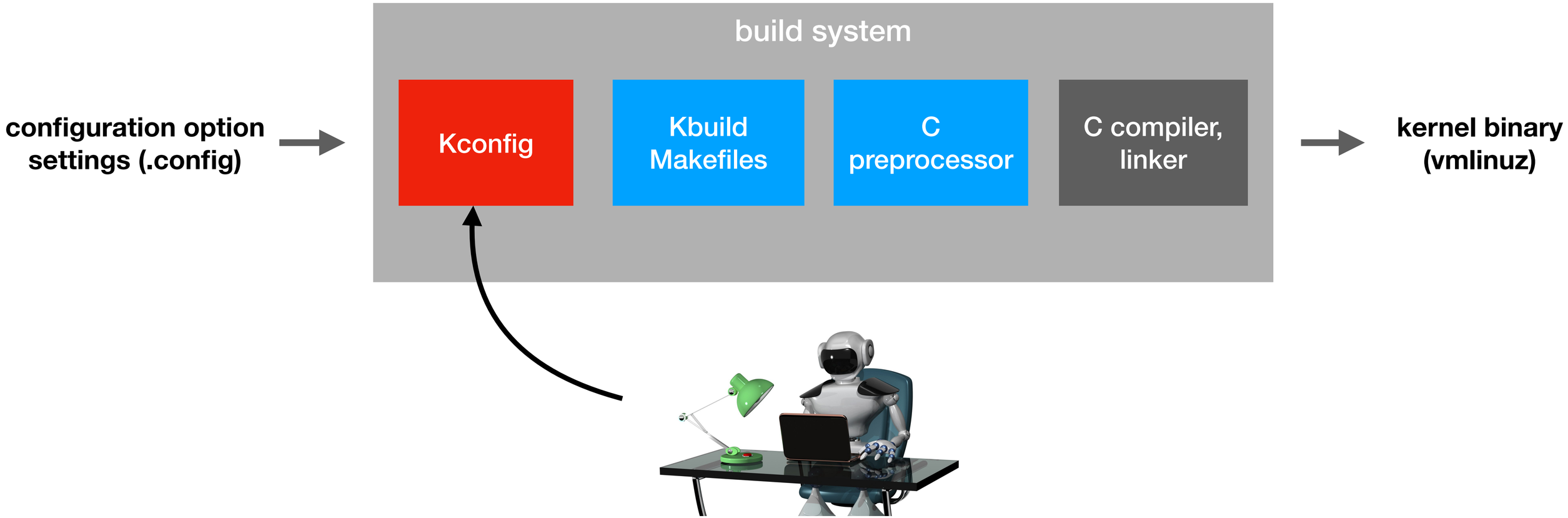


our goal: automatically analyze the build system





today's focus: Kconfig's unmet dependency bugs



how Kconfig works

kconfig language example

```
config TOUCHSCREEN_ADC
    tristate
    prompt "Generic ADC based touchscreen"
    depends on IIO && INPUT_TOUCHSCREEN
    select IIO_BUFFER_CB
```

kconfig language example

declaring the option

```
config TOUCHSCREEN_ADC
	tristate
	prompt "Generic ADC based touchscreen"
	depends on IIO && INPUT_TOUCHSCREEN
	select IIO_BUFFER_CB
```

kconfig language example

```
config TOUCHSCREEN_ADC
	tristate
	prompt "Generic ADC based touchscreen"
	depends on IIO && INPUT_TOUCHSCREEN
	select IIO_BUFFER_CB
```

giving it a type

kconfig language example

```
config TOUCHSCREEN_ADC
    tristate
    prompt "Generic ADC based touchscreen"
    depends on IIO && INPUT_TOUCHSCREEN
    select IIO_BUFFER_CB
```

text description for the user interface

kconfig language example

```
config TOUCHSCREEN_ADC
    tristate
    prompt "Generic ADC based touchscreen"
    depends on IIO && INPUT_TOUCHSCREEN
    select IIO_BUFFER_CB
```

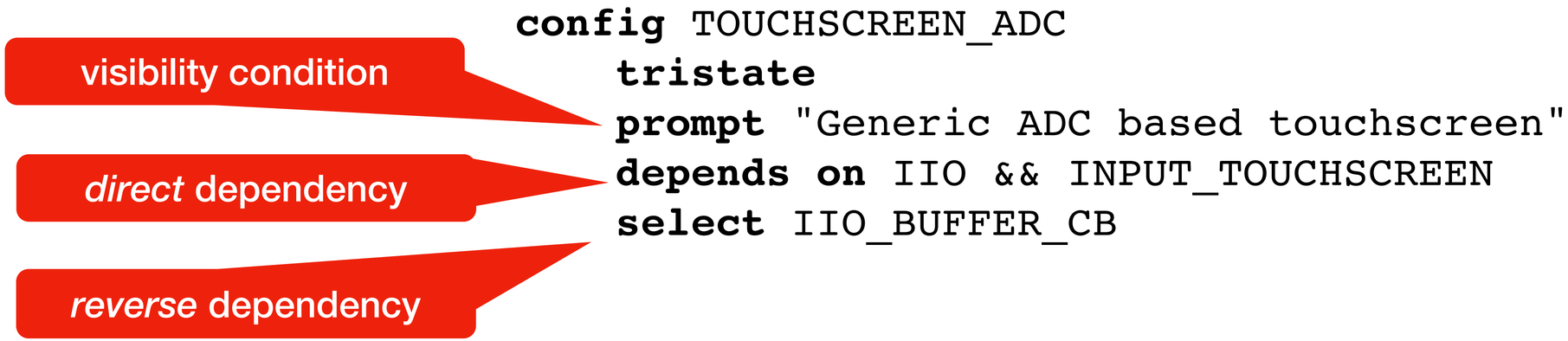
requires IIO and INPUT_TOUCHSCREEN

kconfig language example

```
config TOUCHSCREEN_ADC
    tristate
    prompt "Generic ADC based touchscreen"
    depends on IIO && INPUT_TOUCHSCREEN
    select IIO_BUFFER_CB
```

automatically turns on IIO_BUFFER_CB

kconfig language example



the unmet dependency bug

> “**select should be used with care. select will force a symbol to a value without visiting the dependencies.** By abusing select you are able to select a symbol FOO even if FOO depends on BAR that is not set. In general use select only for non-visible symbols (no prompts anywhere) and for symbols with no dependencies. That will limit the usefulness but on the other hand avoid the illegal configurations all over”

https://www.kernel.org/doc/html/latest/kbuild/kconfig-language.html

an unmet dependency bug in the wild

```
config TOUCHSCREEN_ADC
    tristate
    prompt "Generic ADC based touchscreen"
    depends on IIO && INPUT_TOUCHSCREEN
    select IIO_BUFFER_CB

config IIO_BUFFER
    bool
    prompt "Enable buffer support within IIO"
    depends on IIO

config IIO_BUFFER_CB
    tristate
    prompt "IIO callback buffer"
    depends on IIO && IIO_BUFFER
```

an unmet dependency bug in the wild

```
config TOUCHSCREEN_ADC
    tristate
    prompt "Generic ADC based touchscreen"
    depends on IIO && INPUT_TOUCHSCREEN
    select IIO_BUFFER_CB

config IIO_BUFFER
    bool
    prompt "Enable buffer support within IIO"
    depends on IIO

config IIO_BUFFER_CB
    tristate
    prompt "IIO callback buffer"
    depends on IIO && IIO_BUFFER
```

IIO_BUFFER_CB
depends on
IIO and IIO_BUFFER

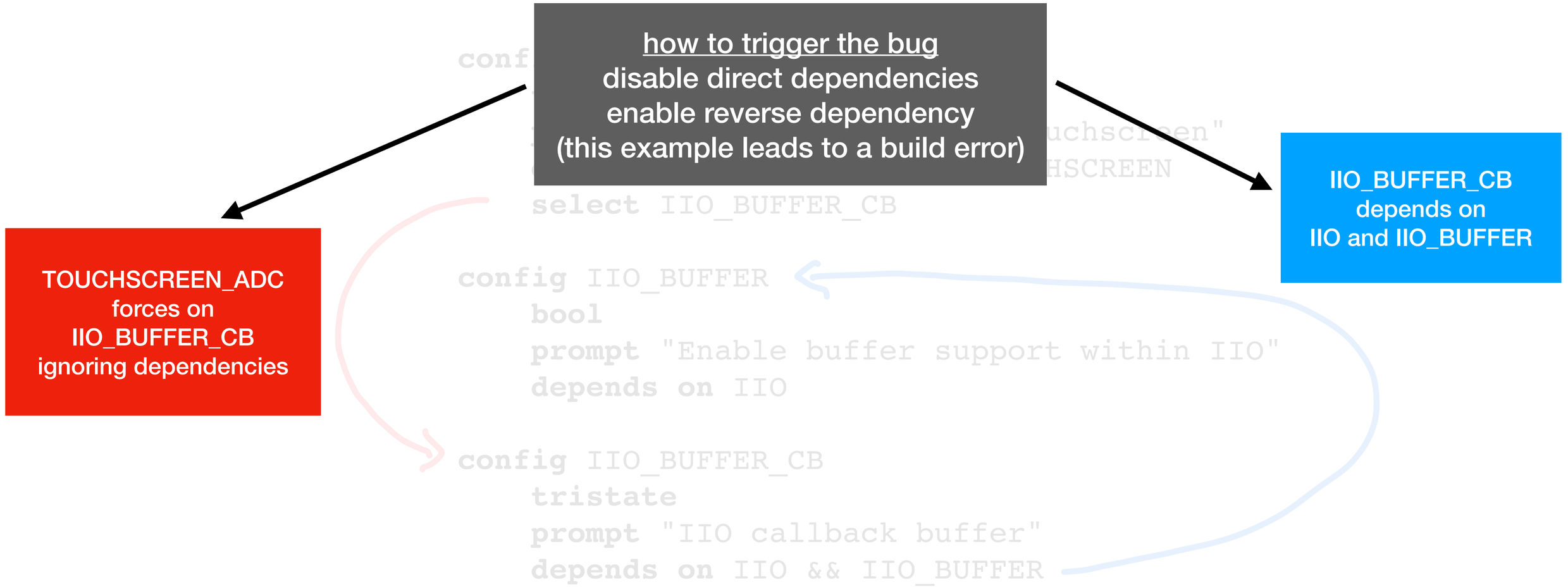
an unmet dependency bug in the wild

```
config TOUCHSCREEN_ADC
    tristate
    prompt "Generic ADC based touchscreen"
    depends on IIO && INPUT_TOUCHSCREEN
    select IIO_BUFFER_CB

config IIO_BUFFER
    bool
    prompt "Enable buffer support within IIO"
    depends on IIO

config IIO_BUFFER_CB
    tristate
    prompt "IIO callback buffer"
    depends on IIO && IIO_BUFFER
```

TOUCHSCREEN_ADC
forces on
IIO_BUFFER_CB
ignoring dependencies

IIO_BUFFER_CB
depends on
IIO and IIO_BUFFER

an unmet dependency bug in the wild



automatically finding unmet dependency bugs

turn “are there unmet dependency bugs?”
into a Boolean satisfiability problem.

use an off-the-shelf solver to get an answer

we first model Kconfig in symbol logic

```
config TOUCHSCREEN_ADC
    tristate
    prompt "Generic ADC based touchscreen"
    depends on IIO && INPUT_TOUCHSCREEN
    select IIO_BUFFER_CB

config IIO_BUFFER
    bool
    prompt "Enable buffer support within IIO"
    depends on IIO

config IIO_BUFFER_CB
    tristate
    prompt "IIO callback buffer"
    depends on IIO && IIO_BUFFER
```

we first model Kconfig in symbol logic

TOUCHSCREEN_ADC implies IIO and INPUT_TOUCHSCREEN

```
config IIO_BUFFER
    bool
    prompt "Enable buffer support within IIO"
    depends on IIO

config IIO_BUFFER_CB
    tristate
    prompt "IIO callback buffer"
    depends on IIO && IIO_BUFFER
```

we first model Kconfig in symbol logic

TOUCHSCREEN_ADC implies IIO and INPUT_TOUCHSCREEN

IIO_BUFFER implies IIO

```
config IIO_BUFFER_CB
    tristate
    prompt "IIO callback buffer"
    depends on IIO && IIO_BUFFER
```

we first model Kconfig in symbol logic

TOUCHSCREEN_ADC implies IIO and INPUT_TOUCHSCREEN

IIO_BUFFER implies IIO

IIO_BUFFER_CB implies (IIO && IIO_BUFFER or TOUCHSCREEN_ADC)

we check every select for an unmet dependency

TOUCHSCREEN_ADC implies IIO and INPUT_TOUCHSCREEN

IIO_BUFFER implies IIO

IIO_BUFFER_CB implies (IIO or TOUCHSCREEN_ADC)

we check every select for an unmet dependency

TOUCHSCREEN_ADC implies IIO and INPUT_TOUCHSCREEN

TOUCHSCREEN_ADC and (IIO and INPUT_TOUCHSCREEN)
and IIO_BUFFER_CB and not (IIO and IIO_BUFFER)

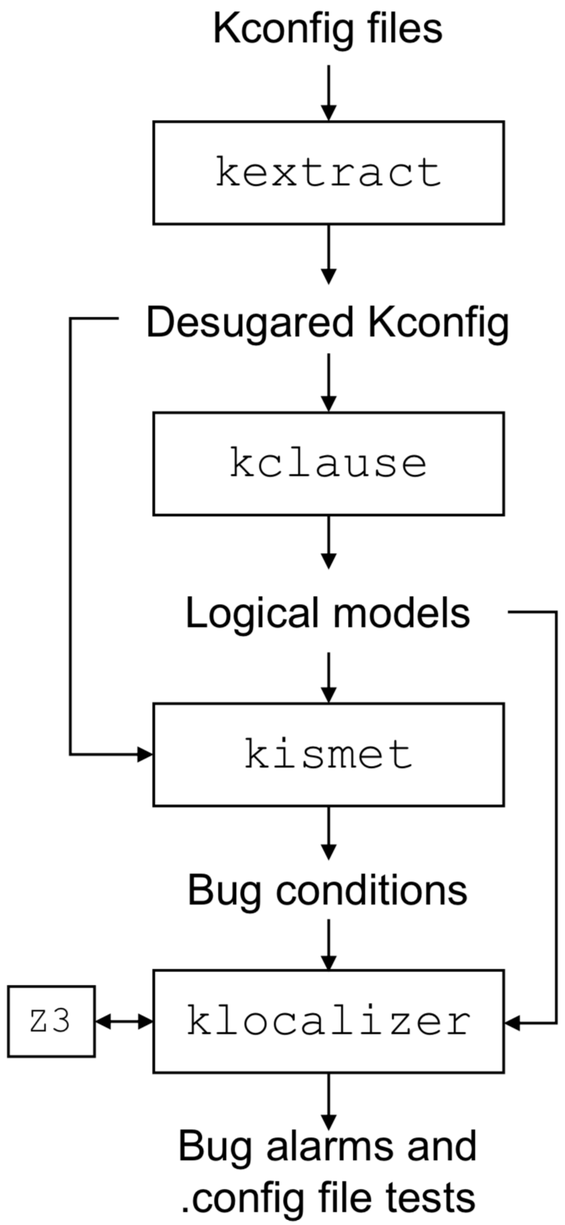
IIO_BUFFER implies IIO

IIO_BUFFER_CB implies (IIO or TOUCHSCREEN_ADC)

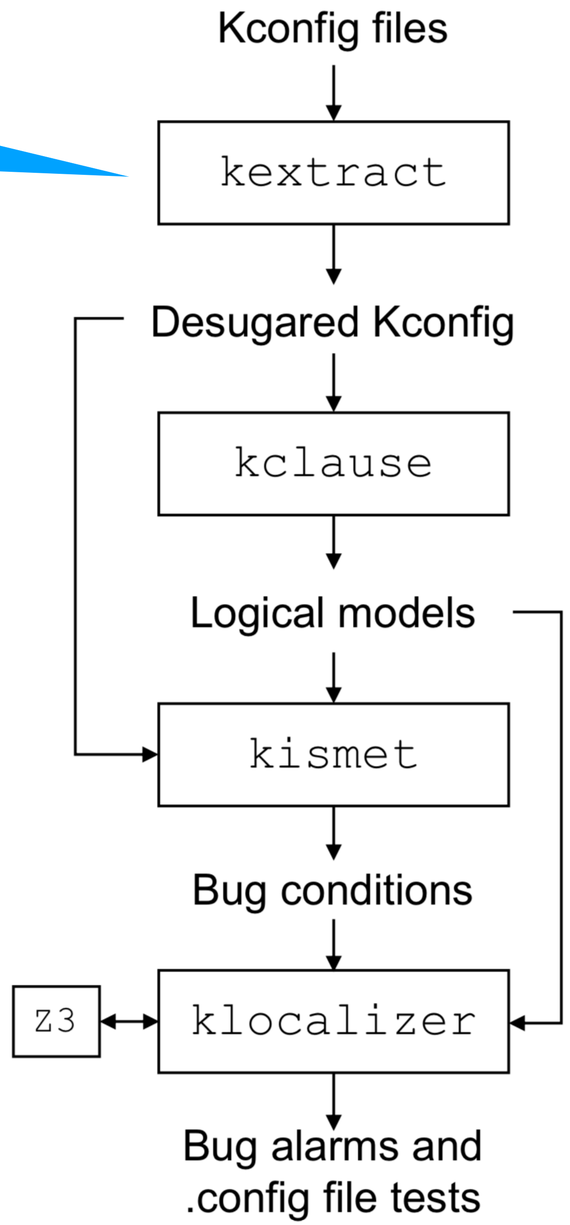
we check every select for an unmet dependency

if this formula is ever true, i.e., SAT,
then an unmet dependency is possible

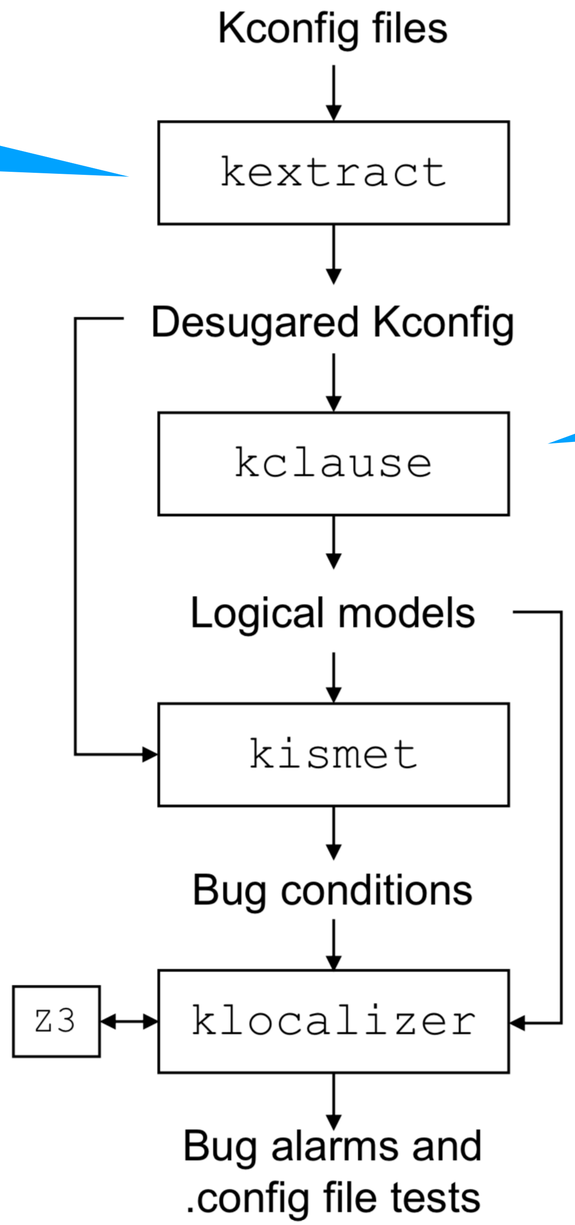
TOUCHSCREEN_ADC and (IIO and INPUT_TOUCHSCREEN)
and IIO_BUFFER_CB and not (IIO and IIO_BUFFER)


Kconfig files
kextract
Desugared Kconfig
kclause
Logical models
kismet
Bug conditions
Z3
klocalizer
Bug alarms and
.config file tests

simplifies syntax, uses kconfig parser from linux source code

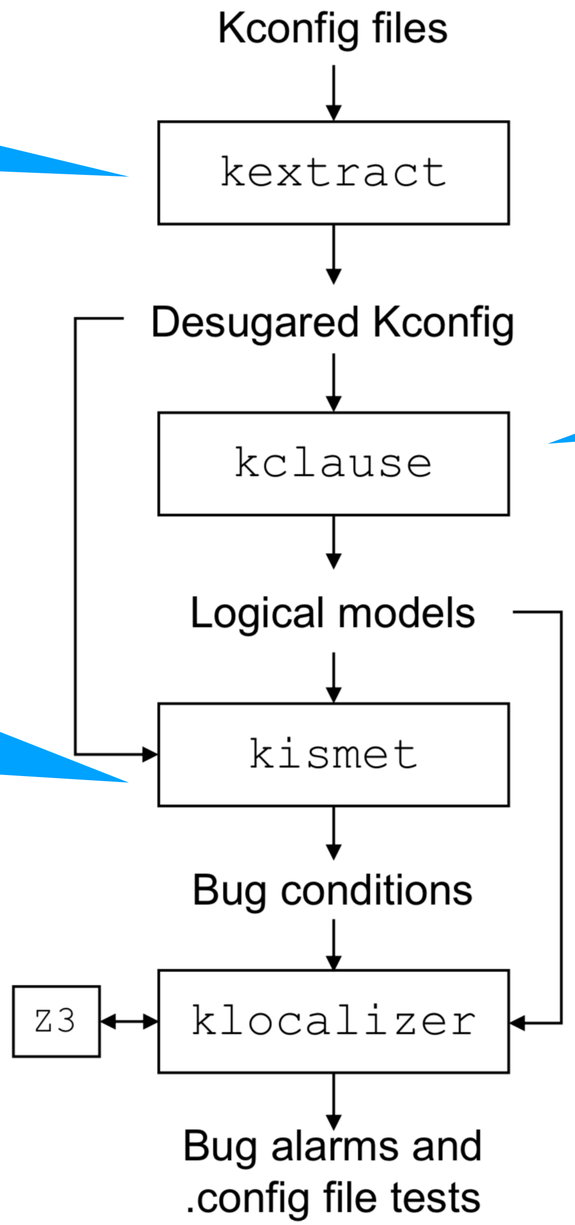


simplifies syntax, uses kconfig parser from linux source code



implements our semantics as translator to logic

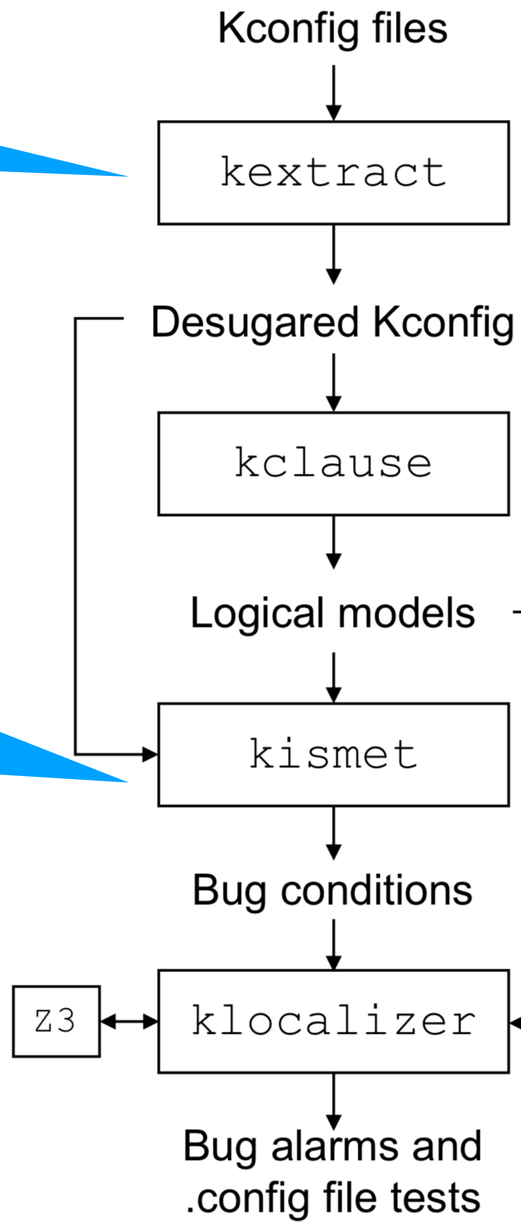
simplifies syntax, uses kconfig parser from linux source code



implements our semantics as translator to logic

generates verification conditions for all select constructs

simplifies syntax, uses kconfig parser from linux source code



implements our semantics as translator to logic

generates verification conditions for all select constructs

dispatches to theorem prover and generates test files on alarm

evaluating our approach

experimental setup

- search for unmet dependency bugs
- linux v5.4.4 source code
- 28 architecture families
- each has its own kconfig specification
  - though most contents are shared between architectures (hardware abstraction layer)
- run kismet on each of the 28
  - deduplicate results from shared select constructs

precision (true positives)

- 151 true positive bugs
  - 781 before deduplicating the 28 kconfig specifications
- bugs validated by automatically generating test configuration files
  - we convert solutions to bug verification condition to linux config file format
- 100% precision
  - for boolean and tristate options
  - we underapproximate non-boolean options

recall (false negatives)

- kismet deliberaly underapproximate for non-boolean options
- unknown ground truth (real-world linux specs)
- we generated and tested about 11,000,000 configuration files
  - used de-facto standard tool, randconfig, over several sequential months of time
- random testing found 8 true positives not found by kismet
  - kismet found 614 not found by randconfig

performance

- 37 to 90 minutes to run a kconfig specification
  - 10,014 to 12,744 select constructs analyzed for each specification
  - all 28 specifications: a little less than one sequential day
  - fast enough to run daily (speed of linux-next repo)
- more bugs found in one hour compared to random testing
  - recall/precision tradeoff for non-boolean underapproximation
  - useful complement to randconfig

impact

- we submitted bug reports (38) and patches for some bugs so far
  - limited by manual effort to patch, report, and converse with maintainers
- all reports (so far) accepted as true
  - at least one known and left in intentionally
  - some report still pending reply or resolution
- 15 patches mainlined so far

tool demo

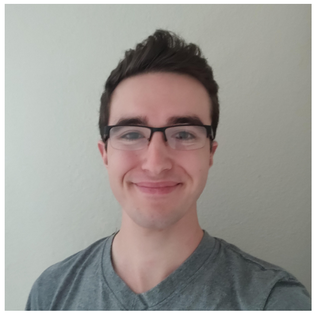
conclusion

student members of project team

Jeho Oh (UT Austin)

Necip Fazıl Yıldıran (UCF)

Julian Braha (UCF)



conclusion

- configurability is great for reuse, a challenge for maintenance
- the build and configuration system is big and complicated
- our goal is automate analyzing the build system components
- kclause models Kconfig in symbolic logic
- kismet uses this model to find unmet dependency bugs

supported by NSF CCF-1941816 and CCF-1840934

try it out!
https://github.com/paulgazz/kmax