The Application of Propositional Logic

Zhaoguo Wang

SAT-Based ATPG. In: Test Pattern Generation using Boolean
Proof Engines. Springer, Dordrecht.

Propositional Logic + SAT

1.1 Propositional Logic

Step 0. Proof.

Step 1. Convert program into mathematical formula.

Step 2. Ask the computer to solve the formula.

1.2 First Order Logic

Step 1. Convert it into first order logic formula.

Step 2. Ask the computer to solve the formula.

1.3 Auto-active Proof

Step 1. Axiom system

Step 2. Ask the computer to check the invariants

PySAT

Outline – This Lecture

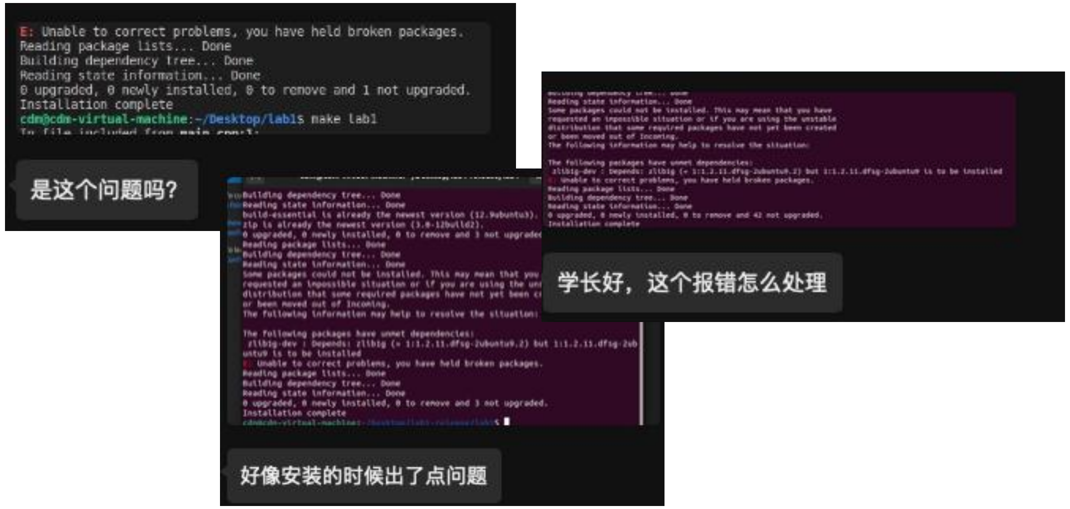
- Package Managers

- Electronic Design Automation (EDA)

Outline – This Lecture

- Package Managers
- Electronic Design Automation (EDA)

Lab1: A Frequently Asked Question



Unsatisfiable

```
The following packages have unmet dependencies:
 zlib1g-dev : Depends: zlib1g (= 1:1.2.11.dfsg-2ubuntu9.2) but 1:1.2.11.dfsg-2ubuntu9 is to be install
ed
E: Unable to correct problems, you have held broken packages.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
0 upgraded, 0 newly installed, 0 to remove and 4 not upgraded.
Installation complete
```

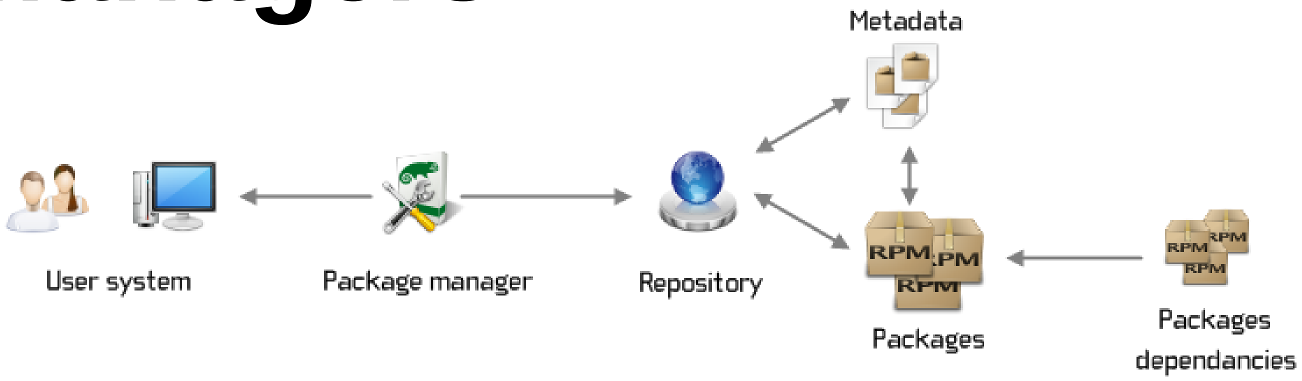
An Example

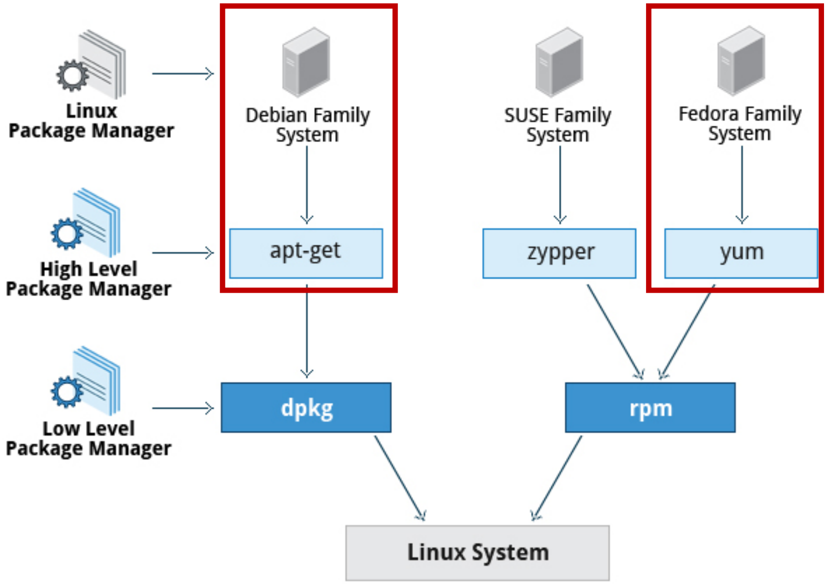
```
Package: firefox-3.0
Priority: optional
Section: web
Installed-Size: 3456
Maintainer: Alexander Sack <asac@ubuntu.com
Architecture: i386
Version: 3.0.16+nobinonly-0ubuntu0.9.04.1
Replaces: firefox (<< 3), firefox-granparadiso, firefox-libt            runk
Provides: firefox-libthai, www-browser
Depends: fontconfig, psmisc, debianutils (>= 1.16), xulrunner-1.9 (>= 1.9.0.1),
    libatk1.0-0 (>= 1.20.0), libc6 (>= 2.4), libcairo2 (>= 1.2.4),
    libfontconfig1 (>= 2.4.0), libfreetype6 (>= 2.2.1), libgcc1 (>= 1:4.1.1),
    libglib2.0-0 (>= 2.16.0), libgtk2.0-0 (>= 2.16.0),
    libnspr4-0d (>= 4.7.3-0ubuntu1~), libpango1.0-0 (>= 1.14.0),
    libstdc++6 (>= 4.1.1),
    firefox-3.0-branding (>= 3.0.3+nobinonly-0ubuntu1~) |
        abrowser-3.0-branding (>= 3.0.3+nobinonly-0ubuntu1~)
Suggests: ubufox,
    firefox-3.0-gnome-support (= 3.0.16+nobinonly-0ubuntu0.9.04.1),
    latex-xft-fonts, libthai0
Conflicts: firefox (<< 3), firefox-granparadiso (<< 3.0~alpha8-0),
    firefox-libthai, firefox-trunk (<< 3.0~a8~cvs20070914t1713-0)
Size: 887822
Description: safe and easy web browser from Mozilla
 Firefox delivers safe, easy web browsing. A familiar user interface,
 enhanced security features including protection from online identity
 and integrated search let you get the most out of the web.
 .
 Install this firefox package too, if you want to be automatically up
 new major firefox versions in the future.
```

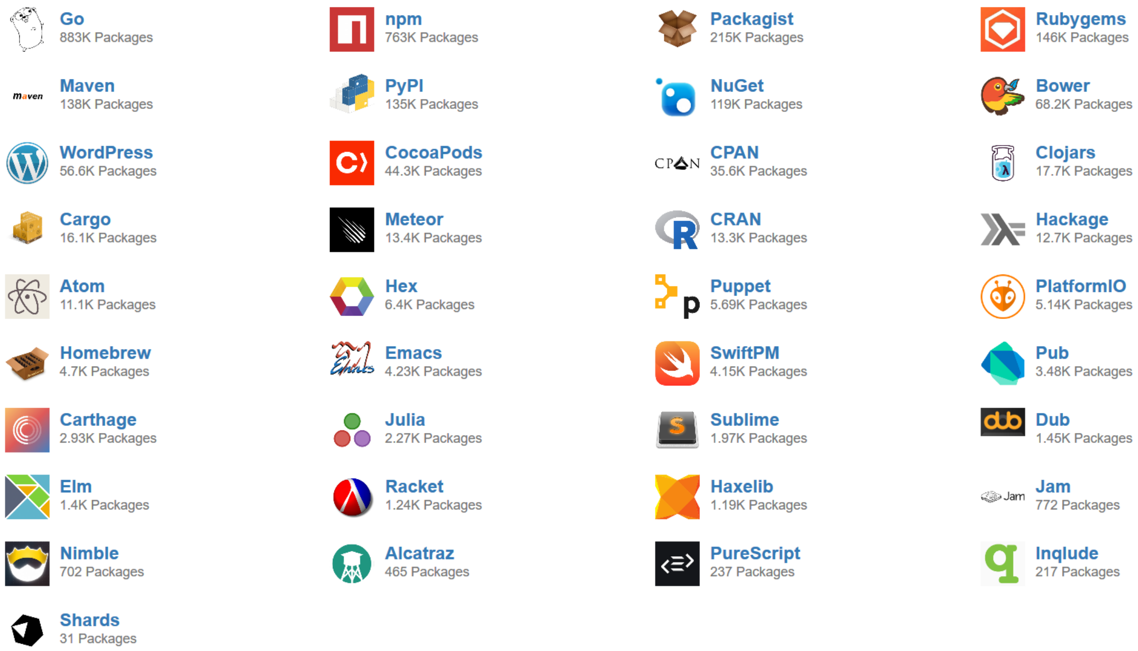
Installing a browser requires to first install some other software in specific versions

There should be no software in these versions

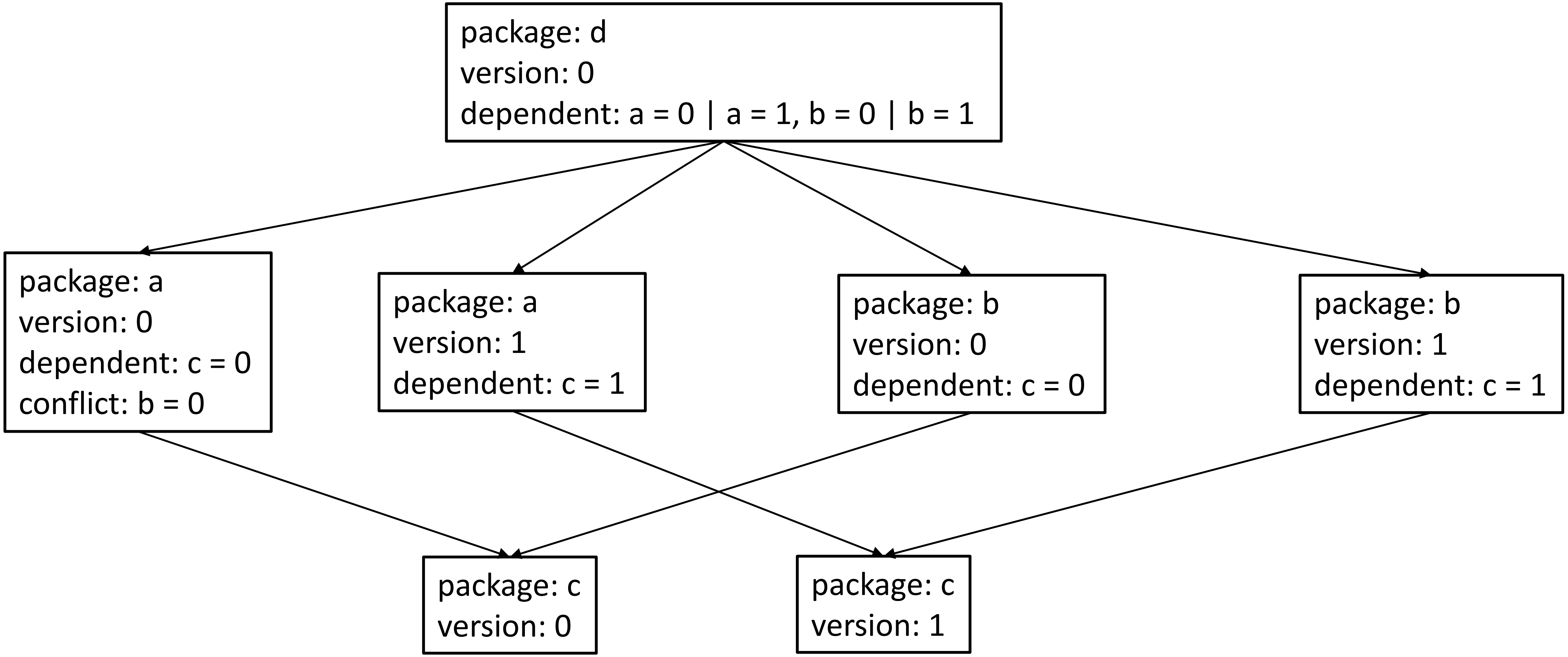
Package Managers







How to Solve Dependencies



Only one version of a package can be installed

Encode Dependencies and Conflict

Dependencies can easily be translated into clauses :

package: d
version: 0
dependent: a = 0 | a = 1, b = 0 | b = 1

$$d_0 \rightarrow ((a_0 \vee a_1) \wedge (b_0 \vee b_1))$$
$$= \neg d_0 \vee ((a_0 \vee a_1) \wedge (b_0 \vee b_1))$$
$$= (\neg d_0 \vee a_0 \vee a_1) \wedge (\neg d_0 \vee b_0 \vee b_1)$$

Conflict can easily be translated into binary clauses :

package: c
version: 0
conflict: c =1

$$c_0 \rightarrow \neg c_1$$
$$= \neg c_0 \vee \neg c_1$$

How to Solve Dependencies

Must install d_0

package: d
version: 0
dependent: a = 0 | a = 1, b = 0 | b = 1

$$\boxed{d_0} \land (\neg d_0 \lor a_0 \lor a_1) \land (\neg d_0 \lor b_0 \lor b_1)$$

package: a
version: 0
dependent: c = 0
conflict: b = 0

$$(\neg a_0 \lor c_0) \land (\neg a_0 \lor \neg b_0)$$

package: a
version: 1
dependent: c = 1

$$(\neg a_1 \lor c_1)$$

package: b
version: 0
dependent: c = 0

$$(\neg b_0 \lor c_0)$$

package: b
version: 1
dependent: c = 1

$$(\neg b_1 \lor c_1)$$

package: c
version: 0

package: c
version: 1

Only one version of a package can be installed： $(\neg a_0 \lor \neg a_1) \land (\neg b_0 \lor \neg b_1) \land (\neg c_0 \lor \neg c_1)$

How to Solve Dependencies

```
> ./apt_dependent.py

SAT: {d0: T, b1: T, c1: T, a1: T, c0: F, a0: F, b0: F}
```

package: d
version: 0
dependent: a = 0 | a = 1, b = 0 | b = 1

Must install d_0

$$\boxed{d_0} \wedge (\neg d_0 \vee a_0 \vee a_1) \wedge (\neg d_0 \vee b_0 \vee b_1)$$

package: a
version: 0
dependent: c = 0
conflict: b = 0

$$(\neg a_0 \vee c_0) \wedge (\neg a_0 \vee \neg b_0)$$

package: a
version: 1
dependent: c = 1

$$(\neg a_1 \vee c_1)$$

package: b
version: 0
dependent: c = 0

$$(\neg b_0 \vee c_0)$$

package: b
version: 1
dependent: c = 1

$$(\neg b_1 \vee c_1)$$

package: c
version: 0

package: c
version: 1

Only one version of a package can be installed： $(\neg a_0 \vee \neg a_1) \wedge (\neg b_0 \vee \neg b_1) \wedge (\neg c_0 \vee \neg c_1)$

Express Multiple Versions

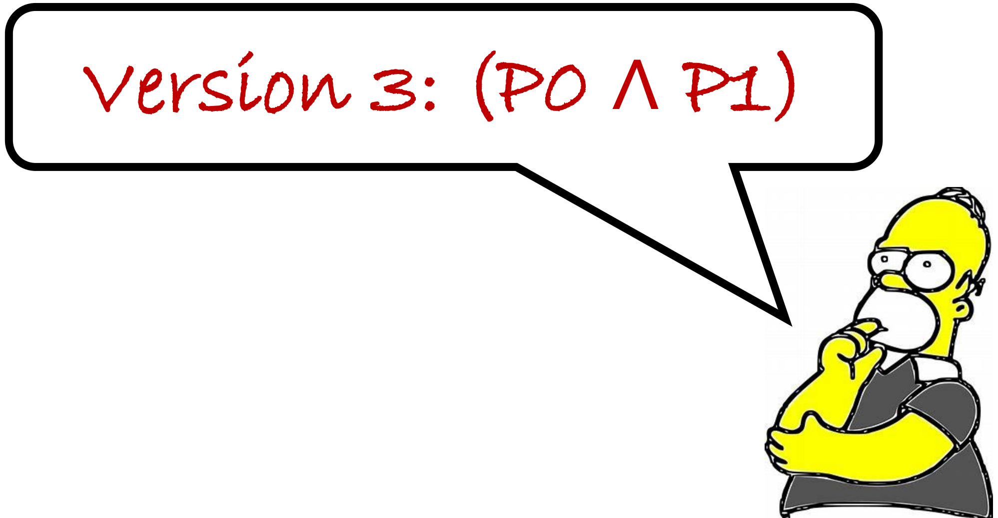
Basic idea : Use multiple variables to encode the version

package: a
version: 0

package: a
version: 1

package: a
version: 2

package: a
version: 3



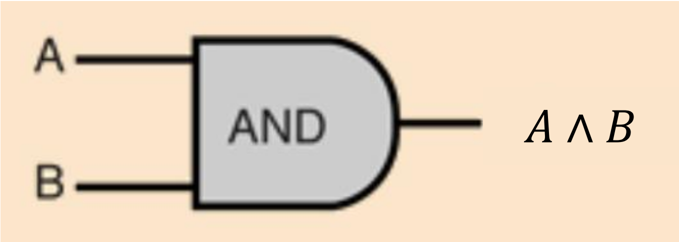
Outline – This Lecture

- Package Managers

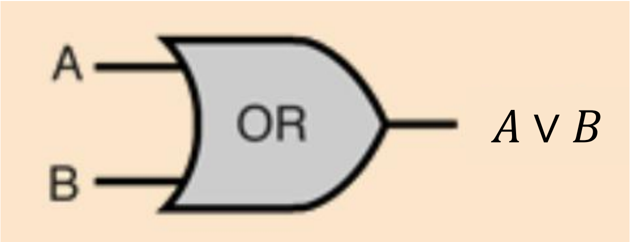
- Electronic Design Automation (EDA)

Integrated Circuits

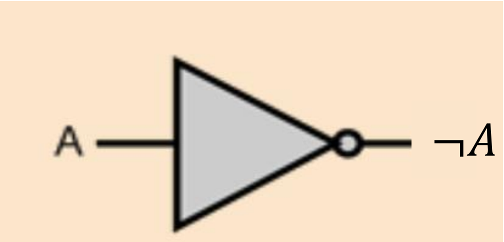
Gates



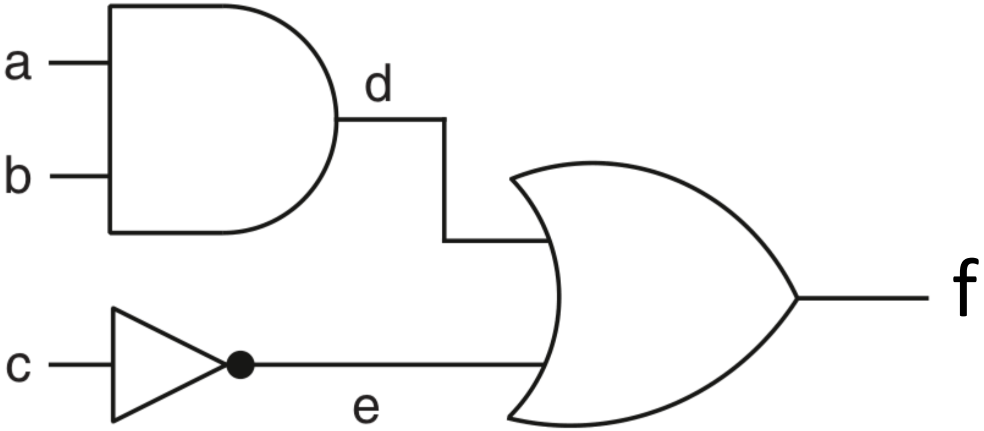
A
B
AND
$A \wedge B$



A
B
OR
$A \vee B$



A
$\neg A$



a
b
d
c
e
f

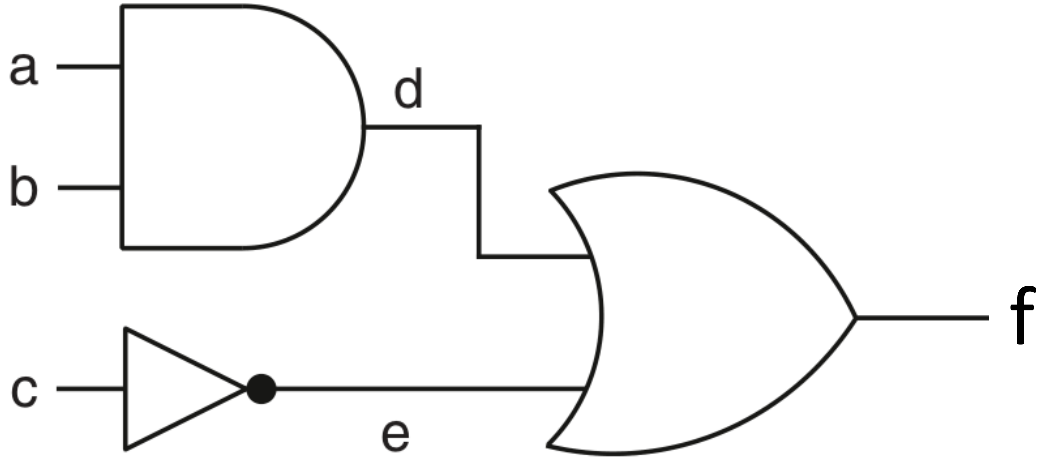
Just like Connectives!

Electronic Design Automation

Use SAT solvers to

- Check the logic of a designed circuit
- Verify if an optimized circuit is equivalent to the original one

Model the Logic of a Circuit

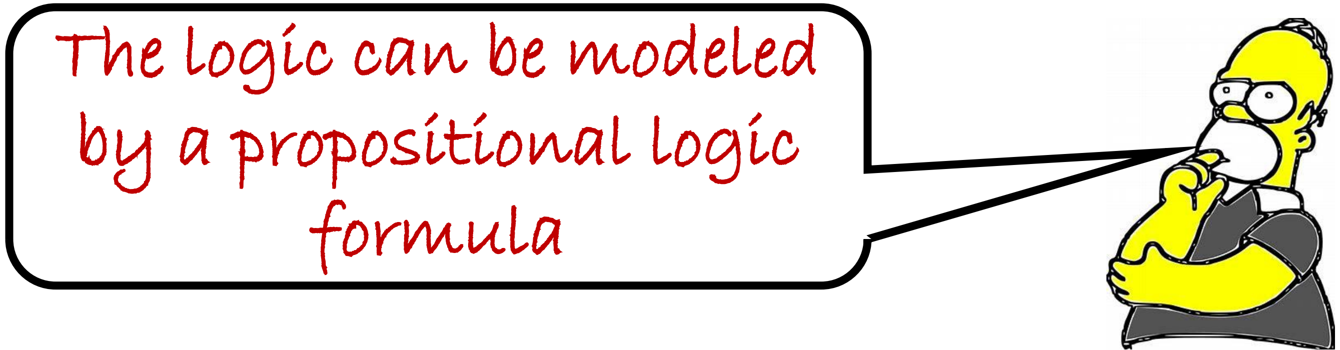


$$\Phi_d = d \leftrightarrow (a \wedge b)$$

$$\Phi_e = e \leftrightarrow \neg c$$

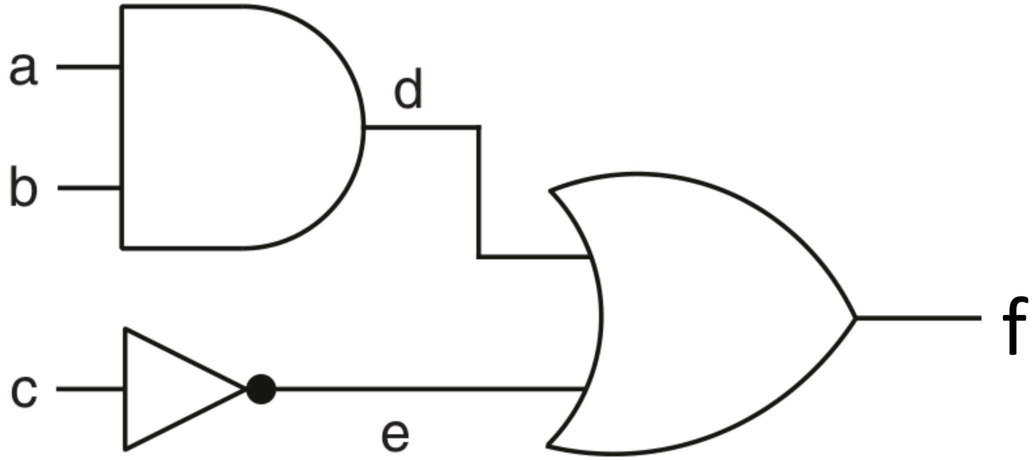
$$\Phi_f = f \leftrightarrow (d \vee e)$$

$$\Phi = \Phi_d \wedge \Phi_e \wedge \Phi_f$$



Check the Logic of a Circuit

Proof Goal: $\neg c \rightarrow f$



$$\Phi_d = d \leftrightarrow (a \land b)$$

$$\Phi_e = e \leftrightarrow \neg c$$

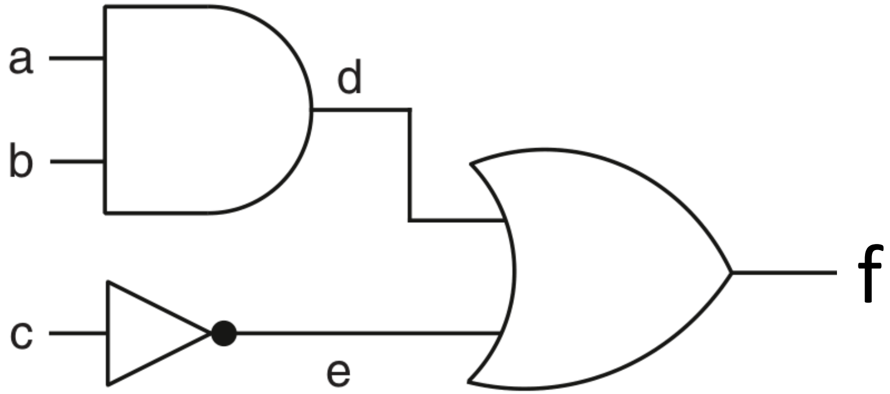
$$\Phi_f = f \leftrightarrow (d \lor e)$$

$$\Phi = \Phi_d \land \Phi_e \land \Phi_f$$

How to check that if c is 0, then f is 1?

Check the Logic of a Circuit

Proof Goal: $\neg c \rightarrow f$



$$\Phi_d = d \leftrightarrow (a \wedge b)$$

$$\Phi_e = e \leftrightarrow \neg c$$

$$\Phi_f = f \leftrightarrow (d \vee e)$$

$$\Phi = \Phi_d \wedge \Phi_e \wedge \Phi_f$$

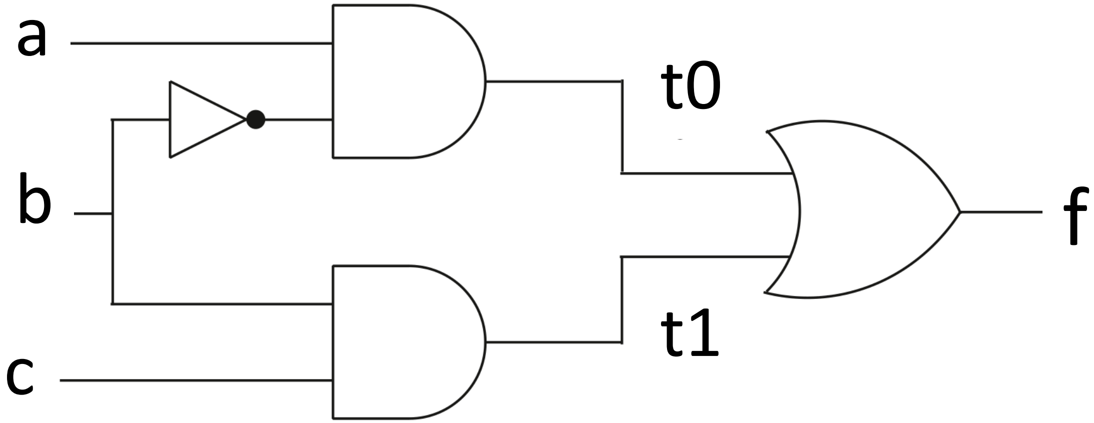
Construct a formula:

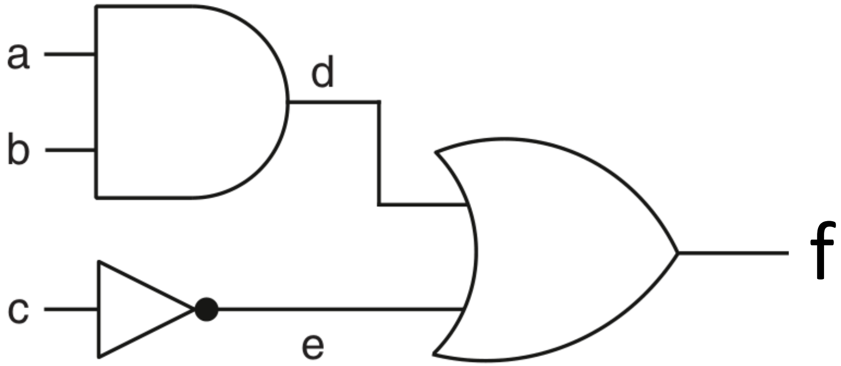
- Logic of the circuit -> Proof Goal

$$\Phi \rightarrow (\neg c \rightarrow f)$$

If the formula is a tautology, the proof goal is true

Verification of Equivalence





$$\Phi_{t_0} = t_0 \leftrightarrow (a \wedge \neg b)$$

$$\Phi_{t_1} = t_1 \leftrightarrow (b \wedge c)$$

$$\Phi_f = f \leftrightarrow (t_0 \vee t_1)$$

$$\Phi_2 = \Phi_{t_0} \wedge \Phi_{t_1} \wedge \Phi_f$$

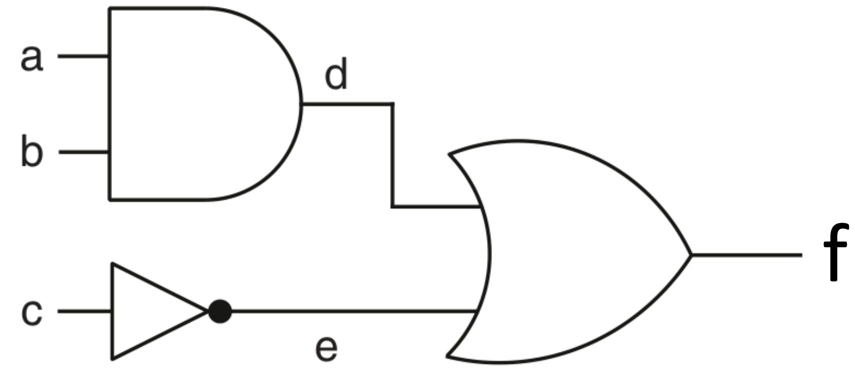
$$\Phi_d = d \leftrightarrow (a \wedge b)$$

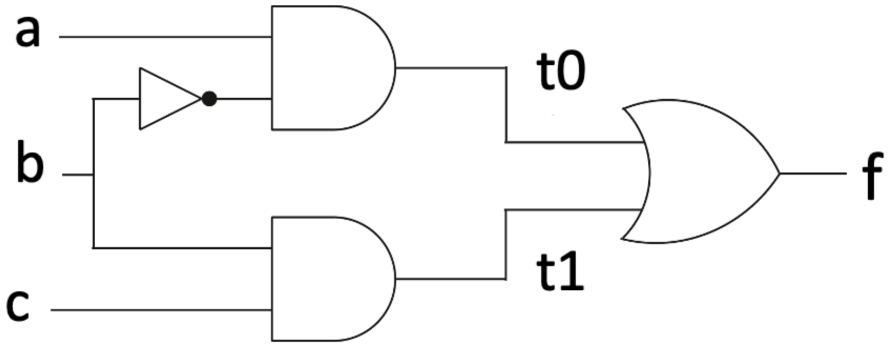
$$\Phi_e = e \leftrightarrow \neg c$$

$$\Phi_f = f \leftrightarrow (d \vee e)$$

$$\Phi_1 = \Phi_d \wedge \Phi_e \wedge \Phi_f$$

Verification of Equivalence





$$\Phi_1 \quad (d \leftrightarrow (a \wedge b)) \wedge (e \leftrightarrow \neg c) \wedge (f \leftrightarrow (d \vee e))$$

$$\Phi_2 \quad \wedge (t_0 \leftrightarrow (a \wedge \neg b)) \wedge (t_1 \leftrightarrow (c \wedge b)) \wedge (f' \leftrightarrow (t_0 \vee t_1))$$

$$\wedge(\neg(f \leftrightarrow f'))$$

If the formula is UNSAT, the two circuits are equivalent

Thanks & Questions