Propositional logic

Short summary of propositional logic

Propositional logic

a formal way for representing complex statements that can be true or false

complex=statements that can be expressed in terms of a number of facts that can be true or false

representing statements + reasoning about them

Boolean formulae in Java

```
int c;
if(a==0)
  c=0;
if(((b==0)&&(d!=0))||(a!=0))
  c=1;
System.out.println(c);
```

contains:

- simple conditions: `a==0`, `b=0` and `d==0`
- negations: `!`
  `a!=0` is the same as `!(a==0)`
- and (conjunction): `&&`
- or (disjunction): `||`

Example of use of logic

Will this program compile?

```
int c;
if(a==0)
  c=0;
if(((b==0)&&(d!=0))||(a!=0))
  c=1;
System.out.println(c);
```

Example: initialization

```
int c;
if(a==0)
  c=0;
if(((b==0)&&(d!=0))||(a!=0))
  c=1;
System.out.println(c);
```

Compiler refuses to compile

Error: `Variable c may not have been initialized`

Meaning: `c` is only initialized within `if` conditional instructions; conditions `might` be false

May be false as far as the compiler knows!

Example: unsatisfiability of conditions

Compiler just assumes that every conditions could be false

In this case:

- if `a` is zero, `c=0` is executed
- if `a` is not zero, `c=1` is executed

No way to make both `a==0` and `(((b==0)&&(d!=0))||(a!=0))` false at the same time

Uses of logic

- check if a formula can be satisfied
- check if a formula is always true
- check if a formula entails another
- ...

Syntax

- variables, like x, y, z...
- connectives: $\wedge$, $\vee$, $\neg$

Examples:

- $(x \vee \neg y) \wedge z$
- $(\neg z \wedge \neg y) \vee (\neg x \wedge (z \wedge x))$
- $(z \vee y \vee \neg x) \wedge w \wedge (x \vee \neg(\neg y \wedge z))$

Variables

Variables can be only true or false

Every elementary condition like **a==0** is expressed by a variable, like x

No way to express what **x** means, just that it is a condition (something that can be true or false)

Semantics

Interpretation = evaluation of the variables

An interpretation I tells the value (true or false) of each variable

Example: $I = \{x{=}true,\ y{=}false,\ z{=}false\}$

Example: $I' = \{x{=}false,\ y{=}false,\ z{=}true\}$

Evaluation: $I \models F$ means that F is true when the variables have the values of I

If $I \models F$, we say that I is a model of F

Reasoning

What can we do in propositional logic?

- checking whether a formula is true according to an interpretation
- checking whether a formula is satisfiable (=it is satisfied by at least an interpretation)
- checking whether a formula is valid (true for all interpretations)
- checking whether a formula implies another formula (F ⇒ G: every model of F is a model of G)

How about the example Java program?

```
int c;
if(a==0)
  c=0;
if(((b==0)&&(d!=0))||(a!=0))
  c=1;
System.out.println(c);
```

`c` is always initialized if `(a==0)&&(((b==0)&&(d!=0))||(a!=0))` is always true

Propositional logic does not work with integers: express `a==0`, `b==0` and `d==0` by x, y and z, respectively

Is $x \wedge ((y \wedge \neg z) \vee \neg x)$ always true?

(yes)

not much useful in practice (just an example)

Other problems can be expressed in propositional logic:

- planning
- scheduling
- diagnosis

CNF form

Definition:

- A propositional CNF formula is a conjunction of clauses
- A clause is a disjunction of literals
- A literal is a variable or the negation of a variable

Example: $(\neg x \lor y \lor z) \land (x \lor \neg z)$

- $\neg x$, y, z, x, $\neg z$ are literals
- $\neg x \lor y \lor z$ is a clause, and so is $x \lor \neg z$
- the whole formula is a conjunction of clauses

Set notation: omit $\land$ by writing the set of clauses:

$\{\neg x \lor y \lor z, x \lor \neg z\}$

CNF: examples

- x (one clause, made of a single positive literal)
- $x \land \neg y$ (two clauses, each made of a single literal, one positive and one literal)
- $(\neg z \lor y \lor w) \land (x \lor y) \land (\neg x \lor z \lor \neg w)$ (three clauses of three, two and three literals respectively)
- $(x \lor y \lor z) \land \neg x \land y \land (w \lor \neg y)$ (four clauses of three, one, one and two literals respectively)

CNF: conversion

Two methods:

- first converts every formula into an equivalent one that is in CNF (transformation may increase size exponentially)
- second converts every formula into an equisatisfiable one that is CNF with at most a polynomial increase of size
  (equisatisfiabile=one is satisfiable if and only if the other one is)

Equivalent conversion

works my "moving" connectives

if a connective is not in the right place:

$\neg$

use De Morgan's laws:

1. $\neg(A \wedge B) = \neg A \vee \neg B$
2. $\neg(A \vee B) = \neg A \wedge \neg B$

$\wedge$ and $\vee$

use distributivity:

1. $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$
2. $A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$

Equivalent conversion: example

$\neg((x\wedge y)\wedge(z\vee\neg(\neg x\vee(z\wedge y))))$

$= \neg(x\wedge y)\vee\neg(z\vee\neg(\neg x\vee(z\wedge y)))$

$= (\neg x\vee\neg y)\vee(\neg z\wedge\neg\neg(\neg x\vee(z\wedge y)))$

$= \neg x\vee\neg y\vee(\neg z\wedge(\neg x\vee(z\wedge y)))$

$= (\neg x\vee\neg y\vee\neg z)\wedge(\neg x\vee\neg y\vee\neg x\vee(z\wedge y))$

$= (\neg x\vee\neg y\vee\neg z)\wedge((\neg x\vee\neg y\vee\neg x\vee z)\wedge(\neg x\vee\neg y\vee\neg x\vee y))$

$= (\neg x\vee\neg y\vee\neg z)\wedge(\neg x\vee\neg y\vee\neg x\vee z)\wedge(\neg x\vee\neg y\vee\neg x\vee y)$

policy:

- push in negation
- push in disjunctions (or, push out conjunctions)

Equivalent conversion: size

in the example, slight increase in formula size

in general: may be exponential

$(x_1\wedge x_2)\vee(x_3\wedge x_4)\vee(x_5\wedge x_6)\vee\ldots\vee(x_{n-1}\wedge x_n)$

$= (x_1\vee(x_3\wedge x_4)\vee(x_5\wedge x_6)\vee\ldots\vee(x_{n-1}\wedge x_n))\wedge(x_2\vee(x_3\wedge x_4)\vee(x_5\wedge x_6)\vee\ldots\vee(x_{n-1}\wedge x_n))$

= repeat for $x_3\wedge x_4$ in both subformulae

= same for $x_5\wedge x_6$ in all four subformulae

= ...

every distribution doubles (more or less) the size of the formula

result is exponential in the number of variables
(all possible disjunctions that contains either x_1 or x_2 and either x_3 or x_4 and…)

Equisatisfiable conversion

employs the connective $\equiv$

$$\boxed{A \equiv B = (A \rightarrow B) \wedge (B \rightarrow A)}$$

can be expressed in terms of $\wedge$, $\vee$ and $\neg$

$$x \equiv (y \wedge z) = (x \rightarrow y) \wedge (x \rightarrow z) \wedge ((y \wedge z) \rightarrow x) = (\neg x \vee y) \wedge (\neg x \vee z) \wedge (\neg y \vee \neg z \vee x)$$
$$x \equiv (y \vee z) = (x \rightarrow (y \vee z)) \wedge ((y \vee z) \rightarrow x) \quad = (\neg x \vee y \vee z) \wedge (\neg y \vee x) \wedge (\neg z \vee x)$$

conversion only needed for these two formulae

Equisatisfiable conversion: example

first push negation to literals (does not increase size)

as above (passages omitted)

$$\neg((x \wedge y) \wedge (z \vee \neg(\neg x \vee (z \wedge y))))$$
$$= \neg x \vee \neg y \vee (\neg z \wedge (\neg x \vee (z \wedge y)))$$

for each subformula (apart literals), define a variable

$$x_1 \equiv \neg z \wedge (\neg x \vee (z \wedge y))$$
$$x_2 \equiv \neg x \vee (z \wedge y)$$
$$x_3 \equiv z \wedge y$$

resulting formula is obtained by:

1. conjoin the original formulae and these three
2. in all of them, replace each topmost subformula with its new variable

in this case, the result is the conjunction of:

- $\neg x \vee \neg y \vee x_1$
- $x_1 \equiv \neg z \wedge x_2$
- $x_2 \equiv \neg x \vee x_3$
- $x_3 \equiv z \wedge y$

Equisatisfiable conversion: size

result **looks** bigger (after converting ≡), but…

$\boxed{\text{conversion increase size only linearly}}$

no repeating doubling-size step, as in the first conversion

↓

Equivalence?

second conversion does not preserve equivalence, but almost

- original formula does not contain x_1, x_2 and x_3
  every value for these variables would do
- the resulting formula contains $x_3 \equiv z \wedge y$
  values $x_3 = false$, $z = true$ and $y = true$ falsifies it

apart from the new variables, models are the same