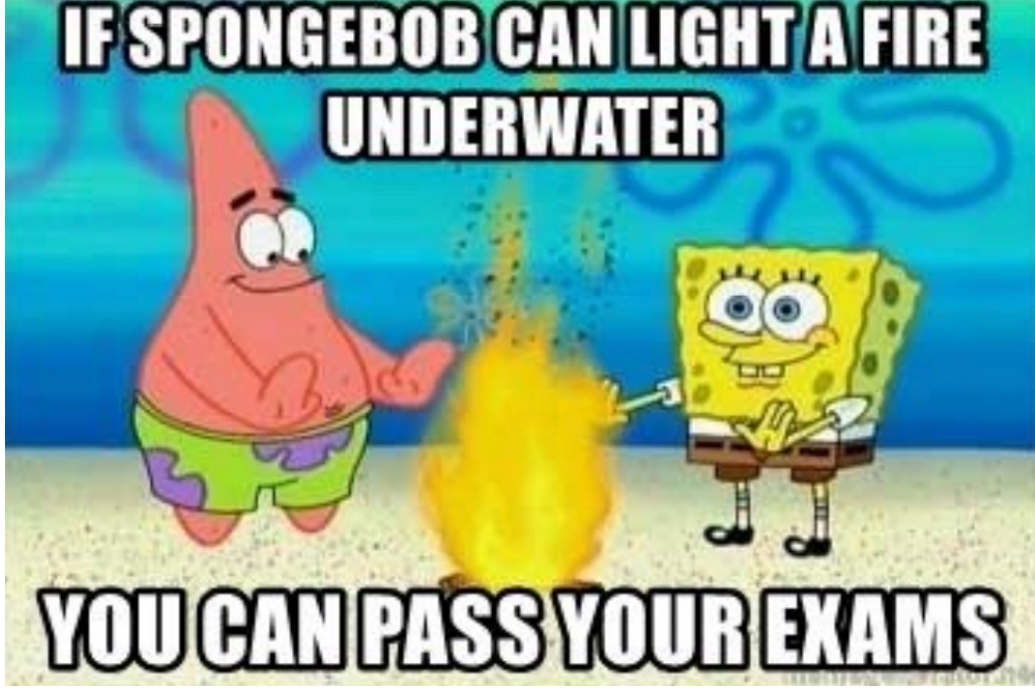
期末复习

Zhaoguo Wang

范围1: 期末考试（闭卷）



- 逻辑1, 2, 4, 5章
  - 不包括存在型前束范式
- 集合论 9, 10, 11.1,11.2 章节

- 一切以课本为准，不要使用DPLL等课本上没有的技术

友情提醒

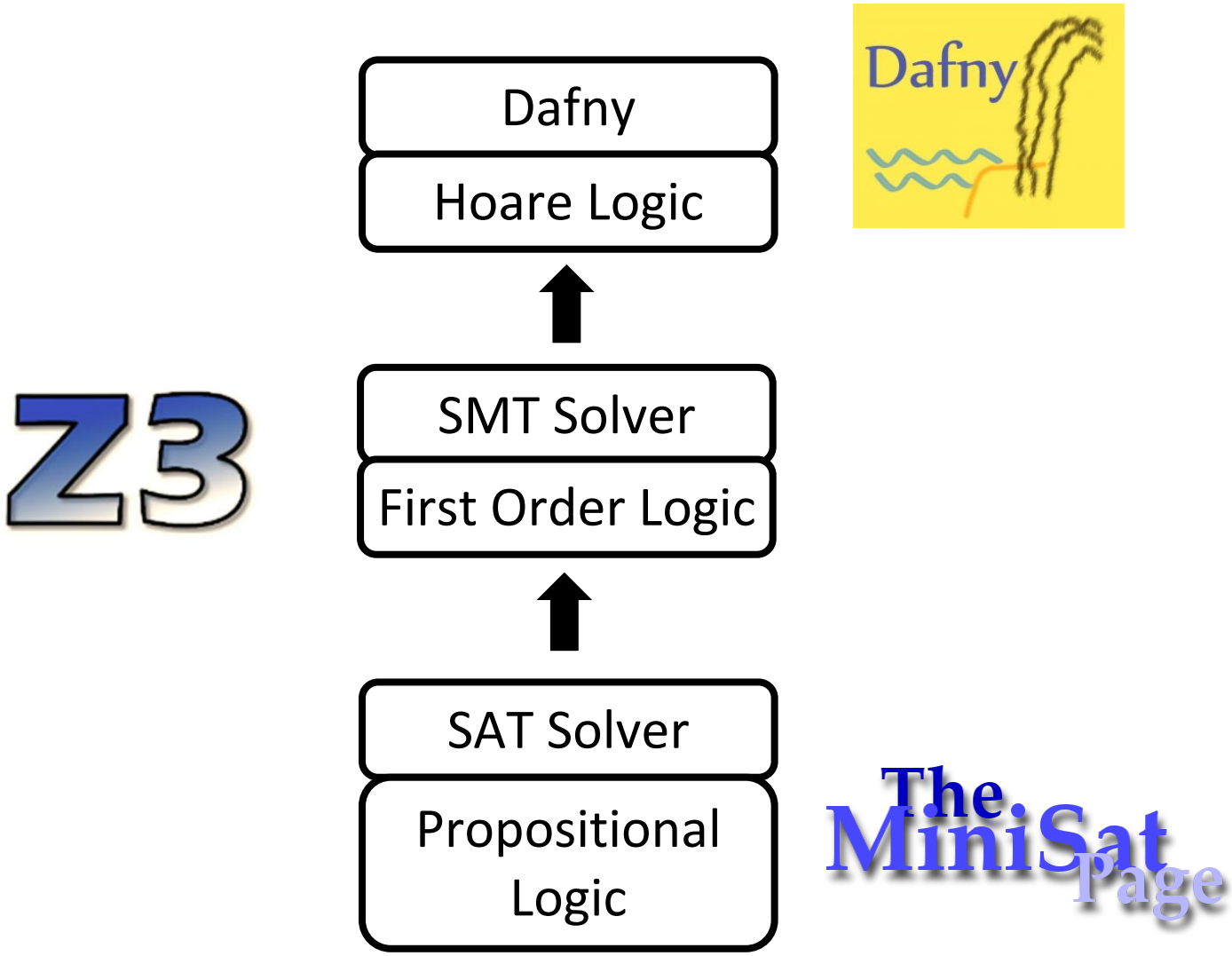
- 可以使用课本的书后习题多加练习
  - 有任何问题欢迎随时联系助教
- 证明题
  - 一定按照课本上的格式证明，不要跳步
  - 不要通过写一大段自然语言进行证明

范围2: quiz（开卷）

- 形式化验证
  - DPLL, convert program into SAT
  - Lazy SMT, EUF, symbolic execution
  - Hoare triple, Hoare inference rules, loop invariant(quiz会给出invariant), weakest precondition
  - 不考WFF的自动识别, strongest postcondition

- 自动机和图灵机
  - DFA, Turing machine basics, the language of Turing machine
  - 不考Turing machine variants, undecidability

Verification Architecture



Part I: Computer Aided Reasoning Overview

1.1 Propositional Logic

Step 1. Convert program into propositional formula.

Step 2. Ask the computer to solve the formula.

1.2 First Order Logic

Step 1. Convert program 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

We start at propositional logic, which is a simple logic. Computer has 1 and 0. Propositional logic has T and F.

Propositional Logic

- A ***proposition(命题)*** is a statement that is, by itself, either true or false.
- Every formula in propositional logic consists of ***propositional variables (命题变项)*** combined via ***propositional connectives (联结词)***.
  - *p, q, …*
  - ¬, ∧, ∨, →, ↔
- Propositions without any connectives are called ***atomic propositions or simple propositions(原子命题/简单命题).***

Well-Formed Formulas (合式公式)

INDUCTIVE DEFINITION of WFF

1). Every single proposition (symbol) is WFF.

2). If *A* and *B* are WFF, so are $(\neg A)$, $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$, and $(A \leftrightarrow B)$.

3). No expression is WFF unless forced by 1) or 2).

- We can omit some brackets.
- All operators are left-associative.
- Operator precedence: $\neg > \wedge > \vee > \rightarrow > \leftrightarrow$

Truth Table (真值表)

- The semantic of a WFF can be represented as a truth table.
- If there are n propositional variables, the truth table has 2^n rows.
- Truth tables and formulas can be converted to each other.

P	Q	$g_0(P,Q)$	$g_1(P,Q)$
F	F	T	T
F	T	T	T
T	F	F	F
T	T	T	F

$g_0(P,Q) = (((\neg P) \wedge (\neg Q)) \vee ((\neg P) \wedge Q)) \vee (P \wedge Q)$

$g_1(P,Q) = ((\neg P) \wedge (\neg Q)) \vee ((\neg P) \wedge Q)$

M1. According to the T rows (DNF)

P	Q	$g_0(P,Q)$	$g_1(P,Q)$
F	F	T	T
F	T	T	T
T	F	F	F
T	T	T	F

$g_0(P,Q) = ((\neg P) \vee Q)$

$g_1(P,Q) = ((\neg P) \vee Q) \wedge ((\neg P) \vee (\neg Q))$

M2. According to the F rows (CNF)

Truth Table

- We can also define new connectives.
- For n propositional variables, we can define $\mathbf{2^{2^n}}$ connectives.
- But some connectives are unnecessary.

P	Q	$P \overline{\vee} Q$
T	T	F
F	T	T
T	F	T
F	F	F

$$P \overline{\vee} Q = ((P \wedge (\neg Q)) \vee ((\neg P) \wedge Q))$$

Completeness of Connectives (完备性)

DEFINITION

A group of connectives are **complete** if

1). Every formula can be converted to an **equivalent** formula and

2). The formula only includes connectives in the group.

- {¬, ∧} and {¬, ∨} are all complete.
- More examples in textbook 2.4.
- We can determine equivalence via truth table or calculation via laws.

Laws

De Morgan's Law (摩根律):

$$\neg(P \wedge Q) = \neg P \vee \neg Q, \neg(P \vee Q) = \neg P \wedge \neg Q$$

Distributive Law (分配律) :

$$P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R)$$
$$P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R)$$

Absorption Law (吸收律) :

$$P \vee (P \wedge Q) = P$$
$$P \wedge (P \vee Q) = P$$

More laws in 2.2

Special Formulas

DEFINITION

Tautology (重言式/永真式) is a proposition which is always true under any interpretation. $P \vee \neg P$

$P \wedge \neg P$

Contradiction (矛盾式) is a proposition which is always false under any interpretation.

If a proposition can be true under some interpretation, it is satisfiable (可满足).

$P3 \wedge \neg(P1 \wedge P2) \wedge \neg(\neg P1 \wedge P3)$

- A is a tautology iff. $\neg A$ is a contradiction (unsatisfiable).
- We can represent a problem as a WFF and prove it by
  - Truth table (very complex because of $\mathbf{2^n}$ rows)
  - SAT solver
  - Deduction

SAT Solver

DEFINITION

In computer science, the satisfiability problem (abbreviated SATISFIABILITY or SAT) is the problem of determining if there exists an interpretation that satisfies a given formula.

- SAT solver is used to solve SAT problem.
- Given a WFF, SAT solver returns “sat” or “unsat”.
- The WFF must be in a normal form (范式).

Normal Form

- A literal（文字）is a propositional variable or its negation.
- A clause（析取式/子句）is a disjunction of one or more literals.
- A conjunctive clause（合取式） is a conjunction of one or more literals.
- Conjunctive normal form (合取范式) $(A \vee \neg B) \wedge (B \vee \neg A) \wedge A$
  - A conjunction of one or more clauses.
- Disjunctive normal form (析取范式) $(A \wedge B) \vee (\neg A \wedge \neg B)$
  - A disjunction of one or more conjunction clauses.

- Every formula can be converted to CNF or DNF.
- SAT solver uses CNF because
  - Converting a formula to DNF may result in a much larger formula.

Principle Normal Form (主范式)

- Given n propositional variables, if every variable exists once in a conjunction clause, the clause is a minterm (极小项)
- Principal Disjunction Normal Form (主析取范式)
  - Disjunction of minterms $g_0(P,Q) = (((\neg P) \wedge (\neg Q)) \vee ((\neg P) \wedge Q)) \vee (P \wedge Q)$

- Given n propositional variables, if every variable exists once in a disjunction clause, the clause is a maxterm (极大项)
- Principal Conjunction Normal Form (主合取范式)
  - Conjunction of maxterms $g_0(P,Q) = ((\neg P) \vee Q)$

- SAT solver do not use PNF because CNF is enough

DPLL

DEFINITION

A literal is defined if its truth value has been guessed or inferred.

OVERVIEW

Iteratively process by following steps:

- Guess the truth value of an undefined literal
- Infer the truth value of other variables with inference rules
- If a clause is false and there is a decision literal, backtrack and re-guess a variable

DPLL

$$\emptyset \parallel 1 \vee 3,\ \overline{2} \vee \overline{4},\ \overline{1} \vee 2 \vee \overline{4},\ \overline{1} \vee 3,\ \overline{2} \vee \overline{3} \vee 4,\ 1 \vee 4,\ \overline{1} \vee 2 \vee 4 \Rightarrow (Decide)$$

$$1^d \parallel 1 \vee 3,\ \overline{2} \vee \overline{4},\ \overline{1} \vee 2 \vee \overline{4},\ \overline{1} \vee 3,\ \overline{2} \vee \overline{3} \vee 4,\ 1 \vee 4,\ \overline{1} \vee 2 \vee 4 \Rightarrow (UnitPropagate)$$

$$1^d 3 \parallel 1 \vee 3,\ \overline{2} \vee \overline{4},\ \overline{1} \vee 2 \vee \overline{4},\ \overline{1} \vee 3,\ \overline{2} \vee \overline{3} \vee 4,\ 1 \vee 4,\ \overline{1} \vee 2 \vee 4 \Rightarrow (Decide)$$

$$1^d 3 2^d \parallel 1 \vee 3,\ \overline{2} \vee \overline{4},\ \overline{1} \vee 2 \vee \overline{4},\ \overline{1} \vee 3,\ \overline{2} \vee \overline{3} \vee 4,\ 1 \vee 4,\ \overline{1} \vee 2 \vee 4 \Rightarrow (UnitPropagate)$$

$$1^d 3 2^d \overline{4} \parallel 1 \vee 3,\ \overline{2} \vee \overline{4},\ \overline{1} \vee 2 \vee \overline{4},\ \overline{1} \vee 3,\ \overline{2} \vee \overline{3} \vee 4,\ 1 \vee 4,\ \overline{1} \vee 2 \vee 4 \Rightarrow (BackTrack)$$

$$1^d 3 \overline{2} \parallel 1 \vee 3,\ \overline{2} \vee \overline{4},\ \overline{1} \vee 2 \vee \overline{4},\ \overline{1} \vee 3,\ \overline{2} \vee \overline{3} \vee 4,\ 1 \vee 4,\ \overline{1} \vee 2 \vee 4 \Rightarrow (UnitPropagate)$$

$$1^d 3 \overline{2}\ \overline{4} \parallel 1 \vee 3,\ \overline{2} \vee \overline{4},\ \overline{1} \vee 2 \vee \overline{4},\ \overline{1} \vee 3,\ \overline{2} \vee \overline{3} \vee 4,\ 1 \vee 4,\ \overline{1} \vee 2 \vee 4 \Rightarrow (BackTrack)$$

$$\overline{1} \parallel 1 \vee 3,\ \overline{2} \vee \overline{4},\ \overline{1} \vee 2 \vee \overline{4},\ \overline{1} \vee 3,\ \overline{2} \vee \overline{3} \vee 4,\ 1 \vee 4,\ \overline{1} \vee 2 \vee 4 \Rightarrow (UnitPropagate)$$

$$\overline{1} 3 \parallel 1 \vee 3,\ \overline{2} \vee \overline{4},\ \overline{1} \vee 2 \vee \overline{4},\ \overline{1} \vee 3,\ \overline{2} \vee \overline{3} \vee 4,\ 1 \vee 4,\ \overline{1} \vee 2 \vee 4 \Rightarrow (UnitPropagate)$$

$$\overline{1} 3 4 \parallel 1 \vee 3,\ \overline{2} \vee \overline{4},\ \overline{1} \vee 2 \vee \overline{4},\ \overline{1} \vee 3,\ \overline{2} \vee \overline{3} \vee 4,\ 1 \vee 4,\ \overline{1} \vee 2 \vee 4 \Rightarrow (UnitPropagate)$$

$$\overline{1} 3 4 \overline{2} \parallel 1 \vee 3,\ \overline{2} \vee \overline{4},\ \overline{1} \vee 2 \vee \overline{4},\ \overline{1} \vee 3,\ \overline{2} \vee \overline{3} \vee 4,\ 1 \vee 4,\ \overline{1} \vee 2 \vee 4$$

Convert Program to WFF

```
void swap(bool& a, bool& b)
{
        a = a ^ b;
        b = b ^ a;
        a = a ^ b;
}
```

a0 b0 } state0

a = a ^ b;

a1 b1 } state1

b = b ^ a;

a2 b2 } state2

a = a ^ b;

a3 b3 } state3

$$(A1 \leftrightarrow (A0 \wedge \neg B0) \vee (\neg A0 \wedge B0)) \wedge$$
$$(B1 \leftrightarrow B0) \wedge$$
$$(A2 \leftrightarrow A1) \wedge$$
$$(B2 \leftrightarrow (A1 \wedge \neg B1) \vee (\neg A1 \wedge B1)) \wedge$$
$$(A3 \leftrightarrow (A2 \wedge \neg B2) \vee (\neg A2 \wedge B2)) \wedge$$
$$(B3 \leftrightarrow B2) \rightarrow$$

Program

$$(A3 \leftrightarrow B0) \wedge (A0 \leftrightarrow B3)$$

Assertion

a and b are swapped iff.
the WFF is a tautology.

Deduction (推理演算)

Prove R when $P \to Q$, $Q \to R$ and P.

(1) P	*introduce premise*
(2) $P \to Q$	*introduce premise*
(3) Q	*Hypothetical reasoning on* (1)(2)
(4) $Q \to R$	*introduce premise*
(5) R	*Hypothetical reasoning on* (3)(4)

前提引入

分离规则

We can also prove a formula by deduction.

Deduction (归结推理法)

For formula $A = P \vee Q$ and $B = \neg P \vee R$, $R(A, B) = Q \vee R$ is a resolvent(归结式) of A and B.

To prove C⟹D, we just need to prove C∧¬D is a contradiction.

Prove that $(P \to Q) \wedge P \Rightarrow Q$.

$$(P \to Q) \wedge P \wedge \neg Q = (\neg P \vee Q) \wedge P \wedge \neg Q$$

$$S = \{\neg P \vee Q, P, \neg Q\}$$

(1) $\neg P \vee Q$	*introduce premise*
(2) P	*introduce premise*
(3) Q	*resolution of* (1) *and* (2)
(4) $\neg Q$	*introduce premise*
(5) $\square$	*resolution of* (3) *and* (4)

Dual Formulas (对偶式)

DEFINITION

Given a formula A, replace ∨, ∧, T, F in A with ∧, ∨, F, T and get A^*. Then A and A^* are dual formulas(对偶式) with each other.

THEOREM

$$A = A(P_1, ..., P_n), \quad A^- = A(\neg P_1, ..., \neg P_n)$$

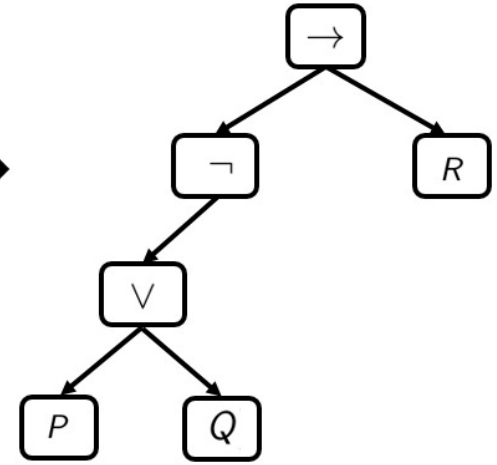
$$\neg(A^*) = (\neg A)^*, \quad \neg(A^-) = (\neg A)^-$$

$$(A^*)^* = A, \quad (A^-)^- = A$$

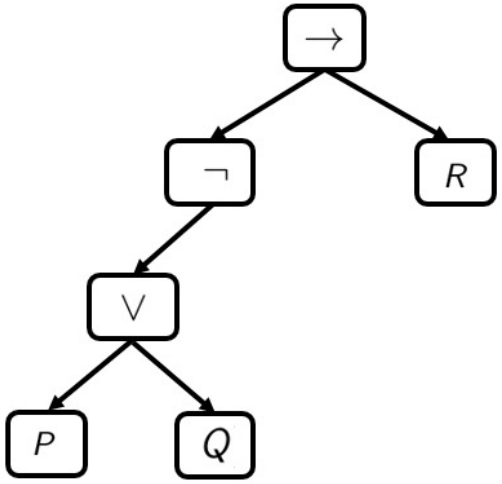
$$(\neg A) = A^{*-}$$

More theorems in 2.5.

Polish Notation and Reverse Polish Notation

$((\neg(P \vee Q)) \to R)$ ➡  ➡ $\to \neg \vee PQR$

Polish Notation
(波兰表达式)

$((\neg(P \vee Q)) \to R)$ ➡  ➡ $PQ \vee \neg R \to$

Reverse Polish
Notation
(逆波兰表达式)

Part I: Computer Aided Reasoning Overview

1.1 Propositional Logic

Step 1. Convert program into propositional formula.

Step 2. Ask the computer to solve the formula.

1.2 First Order Logic

Step 1. Convert program 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

Propositional logic cannot express complex statements conveniently. Then we introduce predicate logic.

Predicate Logic

DEFINITION

Predicates (谓词) describe properties of individuals (个体词).

The range of individuals is domain of discourse (论域).

A function (函数) maps individuals to an individual instead of T or F.

	arguments	results
connective	propositions	a proposition
predicate	individuals	a proposition
function	individuals	an individual

Quantifier (量词)

DEFINITION

A quantifier turns a formula about individuals having some property into a formula about the number (quantity) of individuals having the property.

The blue rectangle is the scope (辖域) of "exists x".

$$(\exists x)((\forall y)(x \leqslant y))$$

x and y are all bound variables (约束变元).

The red rectangle is the scope of "for all y".

Well-formed Formula (合式公式)

INDUCTIVE DEFINITION of WFF

1). Every atomic formula is in WFF.

2). If A and B are WFFs, so are $(\neg \mathrm{A})$, $(\mathrm{A} \wedge \mathrm{B})$, $(\mathrm{A} \vee \mathrm{B})$, $(\mathrm{A} \rightarrow \mathrm{B})$, and $(\mathrm{A} \leftrightarrow \mathrm{B})$.

There is no variable which is bounded in one wff and free in the other wff.

3). If A is a WFF and x is free in A, then $(\forall \mathrm{x})\mathrm{A}$, $(\exists \mathrm{x})\mathrm{A}$ are wffs.

4). No expression is WFF unless forced by 1), 2) or 3).

Decision Problem (判定问题)

DEFINITION

If a formula is true under some interpretation, it is satisfiable (可满足).

If a formula is always false under any interpretations, it is unsatisfiable (不可满足).

If a formula is always true under any interpretations, it is universally valid (普遍有效).

Decision problem in predicate logic is whether there is an effective algorithm to determine if a formula is universally valid.

- Propositional logic is decidable but predicate logic is not decidable
- We can represent a problem as a WFF and prove it by
  - Deduction
  - SMT solver

Deduction (推理规则法)

$Premise:\ \ (\forall x)(P(x) \to Q(x)),\ (\forall x)(Q(x) \to R(x))$

$Conclusion:\ \ (\forall x)(P(x) \to R(x))$

Proof.

$(1)(\forall x)(P(x) \to Q(x))$ $\quad introduce\ premise$

$(2)P(x) \to Q(x)$ $\quad remove\ universal\ quantifier$ 全称量词消去规则

$(3)(\forall x)(Q(x) \to R(x))$ $\quad introduce\ premise$

$(4)Q(x) \to R(x)$ $\quad remove\ universal\ quantifier$

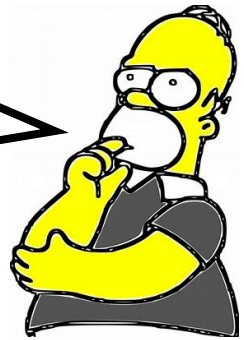
$(5)P(x) \to R(x)$ $\quad syllogism\ on\ (2)(4)$ 三段论

$(6)(\forall x)(P(x) \to R(x))$ $\quad introduce\ universal\ quantifier$

全称量词引入规则

Deduction (归结推理法)

The basic idea is similar to that in propositional logic. But we should eliminate quantifiers firstly.



Prenex Normal Form (前束范式)

DEFINITION

A formula of the predicate calculus is in prenex normal form(PNF) if it is written as a string of quantifiers and bound variables, followed by a quantifier-free part, called the matrix (母式/基式).

$$(Q_1 x_1)\ldots(Q_n x_n)M(x_1, \ldots, x_n)$$

quantifiers

Every formula can be converted into equivalent PNFs. (定理5.3.1)

Equivalence (等値)

$$(\forall x)(P(x) \vee q) = (\forall x)P(x) \vee q$$
$$(\exists x)(P(x) \vee q) = (\exists x)P(x) \vee q$$
$$(\forall x)(P(x) \wedge q) = (\forall x)P(x) \wedge q$$
$$(\exists x)(P(x) \wedge q) = (\exists x)P(x) \wedge q$$
$$(\forall x)(P(x) \to q) = (\exists x)P(x) \to q$$
$$(\exists x)(P(x) \to q) = (\forall x)P(x) \to q$$
$$(\forall x)(p \to Q(x)) = p \to (\forall x)Q(x)$$
$$(\exists x)(p \to Q(x)) = p \to (\exists x)Q(x)$$

分配律

We use laws to convert formulas to PNF.
More laws in 5.1, 5.2.

Prenex Normal Form (前束范式)

$$
\begin{aligned}
&\neg((\forall x)(\exists y)P(a,x,y) \to (\exists x)(\neg(\forall y)Q(y,b) \to R(x)))\\
=&\neg(\neg(\forall x)(\exists y)P(a,x,y) \vee (\exists x)(\neg\neg(\forall y)Q(y,b) \vee R(x)))\\
=&(\forall x)(\exists y)P(a,x,y) \wedge (\forall x)((\exists y)\neg Q(y,b) \wedge \neg R(x))\\
=&(\forall x)((\exists y)P(a,x,y) \wedge (\exists y)\neg Q(y,b) \wedge \neg R(x))\\
=&(\forall x)((\exists y)P(a,x,y) \wedge (\exists z)\neg Q(z,b) \wedge \neg R(x))\\
=&(\forall x)(\exists y)(\exists z)(P(a,x,y) \wedge \neg Q(z,b) \wedge \neg R(x))\\
=&(\forall x)(\exists y)(\exists z)S(a,b,x,y,z)
\end{aligned}
$$

Skolem Normal Form (Skolem标准形)

DEFINITION

A formula is in Skolem normal form if it is in prenex normal form with only universal quantifiers.

THEOREM

Every formula A can be converted to corresponding Skolem normal form B. A is unsatisfiable iff B is unsatisfiable.

This transformation does not preserve the semantics but it preserves satisfiability.

Skolem Normal Form (Skolem标准形)

$$(\forall x)(\exists y)(\exists z)S(a,b,x,y,z)$$

$$(\forall x)(\exists z)S(a,b,x,f(x),z)$$

$$(\forall x)S(a,b,x,f(x),g(x))$$

y depends on x

z depends on x

If x doesn't depend on any variables, we replace x with a constant.

Deduction (归结推理法)

$$(\forall x)(P(x) \to Q(x)) \wedge (\forall x)(Q(x) \to R(x)) \Rightarrow (\forall x)(P(x) \to R(x))$$

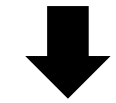
$$(\forall x)(P(x) \to Q(x)) \wedge (\forall x)(Q(x) \to R(x)) \wedge \neg(\forall x)(P(x) \to R(x))$$

$$(\forall x)(P(x) \to Q(x)) \wedge (\forall x)(Q(x) \to R(x)) \wedge (\exists x)(P(x) \wedge \neg R(x))$$

$$(\forall x)(P(x) \to Q(x)) \wedge (\forall x)(Q(x) \to R(x)) \wedge (P(a) \wedge \neg R(a))$$

$$(\forall x)((\neg P(x) \vee Q(x)) \wedge (\neg Q(x) \vee R(x)) \wedge (P(a) \wedge \neg R(a)))$$

Skolem标准形



子句集

$$\neg P(x) \vee Q(x) \qquad \neg Q(x) \vee R(x) \qquad P(a) \qquad \neg R(a)$$

Deduction (归结推理法)

$$\neg P(x) \vee Q(x) \qquad \neg Q(x) \vee R(x) \qquad P(a) \quad \neg R(a)$$

$(1) \neg P(x) \vee Q(x)$

$(2) \neg Q(x) \vee R(x)$

$(3) P(a)$

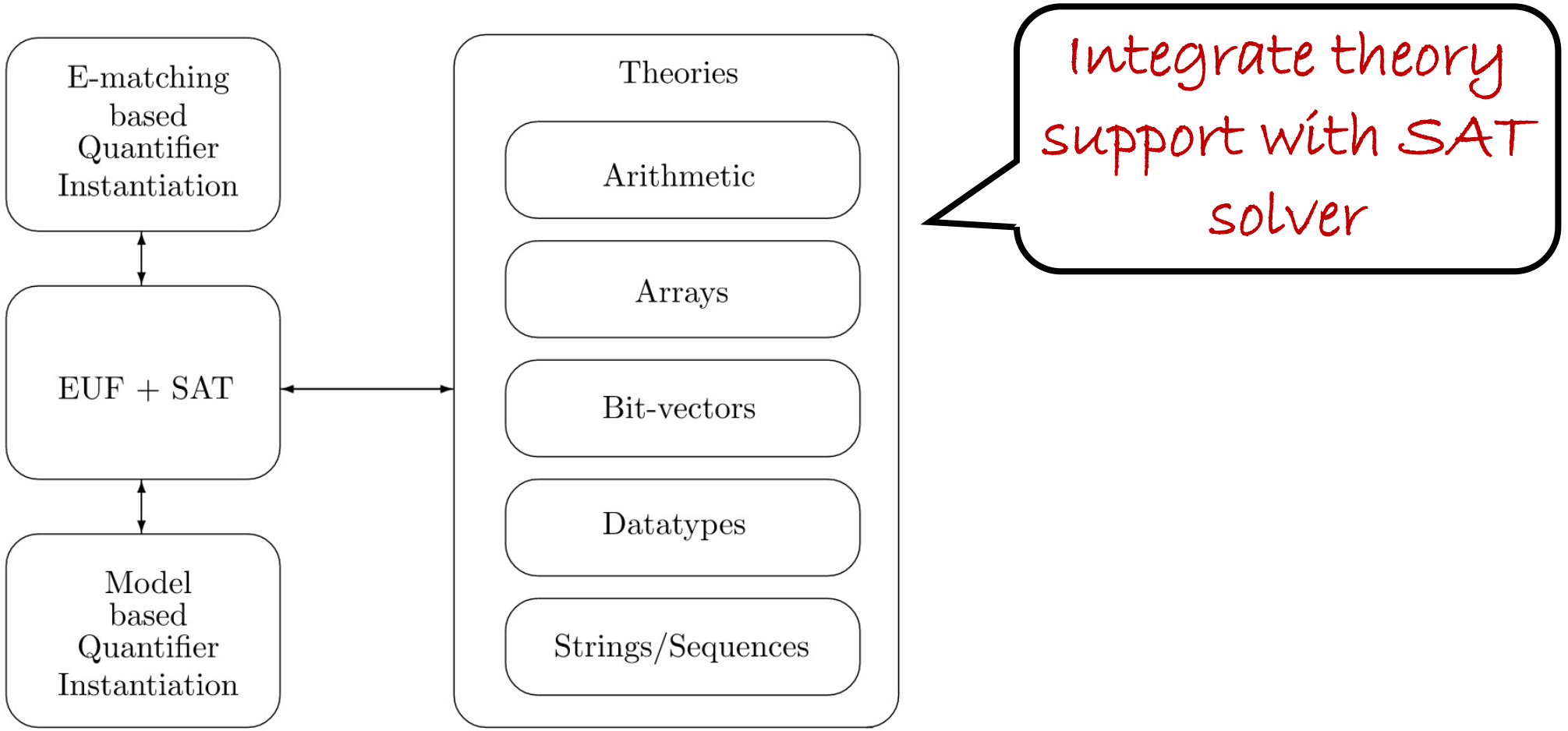
$(4) \neg R(a)$

$(5) Q(a)$ $\quad resolution\ on\ (1)(3)$

$(6) R(a)$ $\quad resolution\ on\ (2)(5)$

$(7) \square$ $\quad resolution\ on\ (4)(6)$

Lazy SMT Techniques



Architecture of Z3's SMT Core solver

Lazy SMT Techniques

$$
\begin{array}{l}
(x = 1 \vee x = 3) \wedge \\
(y = 1 \vee y = 2 \vee y = 3) \wedge \\
(z = 3) \wedge \\
\neg(x = y) \wedge \\
\neg(x = z) \wedge \\
\neg(y = z)
\end{array}
\quad \rightarrow \quad
\begin{array}{l}
(p1 \vee p2) \wedge \\
(p3 \vee p4 \vee p5) \wedge \\
(p6) \wedge \\
\neg p7 \wedge \\
\neg p8 \wedge \\
\neg p9
\end{array}
$$

Step1: replace atomic formulas with fresh propositional variables.

Lazy SMT Techniques

$$(x = 1 \vee x = 3) \wedge$$
$$(y = 1 \vee y = 2 \vee y = 3) \wedge$$
$$(z = 3) \wedge$$
$$\neg(x = y) \wedge$$
$$\neg(x = z) \wedge$$
$$\neg(y = z)$$

$$(p1 \vee p2) \wedge$$
$$(p3 \vee p4 \vee p5) \wedge$$
$$(p6) \wedge$$
$$\neg p7 \wedge$$
$$\neg p8 \wedge$$
$$\neg p9$$

DPLL

UNSAT

If it is unsatisfiable, the original formula is also unsatisfiable.

Step2: use SAT solvers to get assignments.

Lazy SMT Techniques

$$
\begin{aligned}
&(x = 1 \vee x = 3) \wedge \\
&(y = 1 \vee y = 2 \vee y = 3) \wedge \\
&(z = 3) \wedge \\
&\neg(x = y) \wedge \\
&\neg(x = z) \wedge \\
&\neg(y = z)
\end{aligned}
\quad \rightarrow \quad
\begin{aligned}
&(p1 \vee p2) \wedge \\
&(p3 \vee p4 \vee p5) \wedge \\
&(p6) \wedge \\
&\neg p7 \wedge \\
&\neg p8 \wedge \\
&\neg p9
\end{aligned}
$$

DPLL

$$
\begin{aligned}
&p1 = T, p2 = F, p3 = T, p4 = F, \\
&p5 = F, p6 = T, p7 = F, p8 = F, \\
&p9 = F
\end{aligned}
$$

DPLL will not give this result. But to describe lazy SMT, we suppose this is the result.

Step2: use SAT solvers to get assignments.

Lazy SMT Techniques

$$
\begin{aligned}
&(x = 1 \vee x = 3) \wedge \\
&(y = 1 \vee y = 2 \vee y = 3) \wedge \\
&(z = 3) \wedge \\
&\neg(x = y) \wedge \\
&\neg(x = z) \wedge \\
&\neg(y = z)
\end{aligned}
\quad \rightarrow \quad
\begin{aligned}
&(p1 \vee p2) \wedge \\
&(p3 \vee p4 \vee p5) \wedge \\
&(p6) \wedge \\
&\neg p7 \wedge \\
&\neg p8 \wedge \\
&\neg p9
\end{aligned}
$$

DPLL →

$$
\begin{aligned}
&p1 = T, p2 = F, p3 = T, p4 = F, \\
&p5 = F, p6 = T, p7 = F, p8 = F, \\
&p9 = F
\end{aligned}
$$

$$
\begin{aligned}
&x = 1, x \neq 3, y = 1, y \neq 2, \\
&y \neq 3, z = 3, x \neq y, x \neq z, \\
&y \neq z
\end{aligned}
$$

FALSE

Step3: use T-solvers to check the satisfiability.

Lazy SMT Techniques

$$(x = 1 \vee x = 3) \wedge$$
$$(y = 1 \vee y = 2 \vee y = 3) \wedge$$
$$(z = 3) \wedge$$
$$\neg(x = y) \wedge$$
$$\neg(x = z) \wedge$$
$$\neg(y = z)$$

$$(p1 \vee p2) \wedge$$
$$(p3 \vee p4 \vee p5) \wedge$$
$$(p6) \wedge$$
$$\neg p7 \wedge$$
$$\neg p8 \wedge$$
$$\neg p9$$

DPLL

$$p1 = T, p2 = F, p3 = T, p4 = F,$$
$$p5 = F, p6 = T, p7 = F, p8 = F,$$
$$p9 = F$$

$$\neg(p1 \wedge \neg p2 \wedge p3 \wedge \neg p4 \wedge$$
$$\neg p5 \wedge p6 \wedge \neg p7 \wedge \neg p8 \wedge$$
$$\neg p9)$$

$$x = 1, x \neq 3, y = 1, y \neq 2,$$
$$y \neq 3, z = 3, x \neq y, x \neq z,$$
$$y \neq z$$

FALSE

If it is unsatisfiable, tell SAT solver to give another assignment.

Lazy SMT Techniques

$$
\begin{aligned}
&(x = 1 \vee x = 3) \wedge \\
&(y = 1 \vee y = 2 \vee y = 3) \wedge \\
&(z = 3) \wedge \\
&\neg(x = y) \wedge \\
&\neg(x = z) \wedge \\
&\neg(y = z)
\end{aligned}
\quad \rightarrow \quad
\begin{aligned}
&(p1 \vee p2) \wedge \\
&(p3 \vee p4 \vee p5) \wedge \\
&(p6) \wedge \\
&\neg p7 \wedge \\
&\neg p8 \wedge \\
&\neg p9
\end{aligned}
$$

DPLL

$$
\begin{aligned}
&p1 = T, p2 = F, p3 = F, p4 = T \\
&p5 = F, p6 = T, p7 = F, p8 = F, \\
&p9 = F
\end{aligned}
$$

$$
\begin{aligned}
&\neg(p1 \wedge \neg p2 \wedge p3 \wedge \neg p4 \wedge \\
&\neg p5 \wedge p6 \wedge \neg p7 \wedge \neg p8 \wedge \\
&\neg p9)
\end{aligned}
$$

$$
\begin{aligned}
&x = 1, x \neq 3, y = 1, y \neq 2, \\
&y \neq 3, z = 3, x \neq y, x \neq z, \\
&y \neq z
\end{aligned}
$$

FALSE

Lazy SMT Techniques

$$(x = 1 \vee x = 3) \wedge$$
$$(y = 1 \vee y = 2 \vee y = 3) \wedge$$
$$(z = 3) \wedge$$
$$\neg(x = y) \wedge$$
$$\neg(x = z) \wedge$$
$$\neg(y = z)$$

$$(p1 \vee p2) \wedge$$
$$(p3 \vee p4 \vee p5) \wedge$$
$$(p6) \wedge$$
$$\neg p7 \wedge$$
$$\neg p8 \wedge$$
$$\neg p9$$

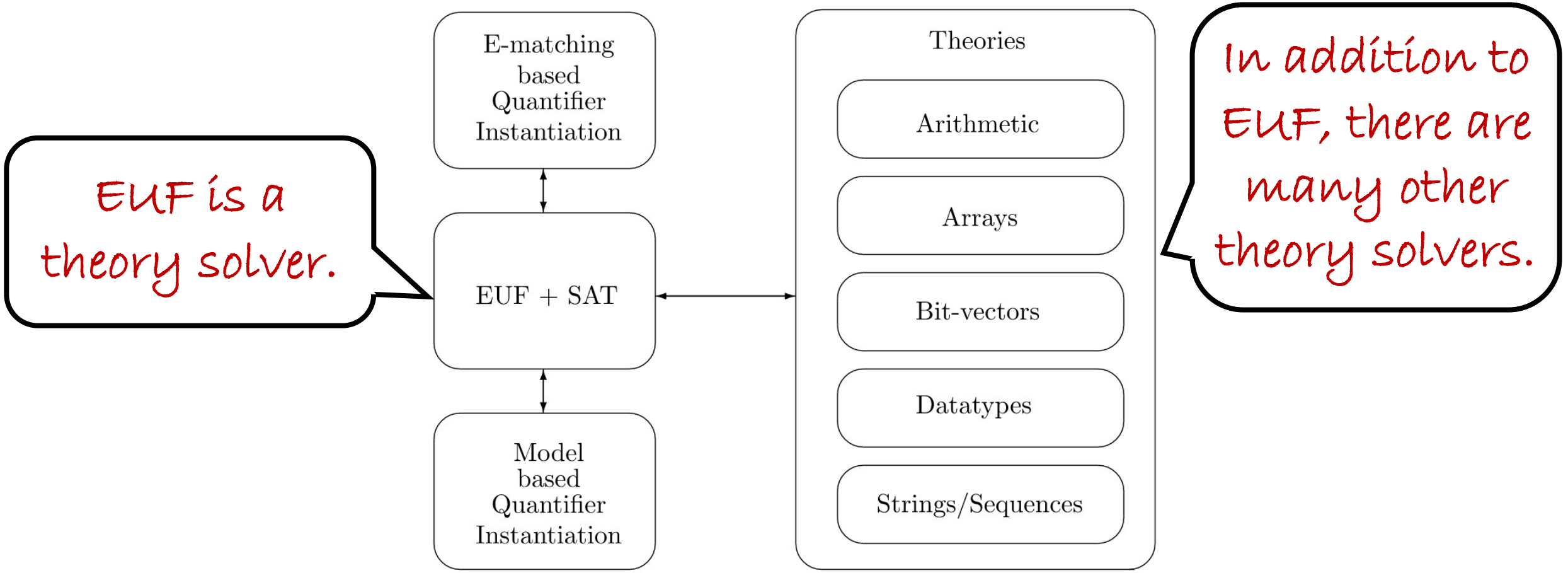
DPLL

$$\neg(p1 \wedge \neg p2 \wedge p3 \wedge \neg p4 \wedge$$
$$\neg p5 \wedge p6 \wedge \neg p7 \wedge \neg p8 \wedge$$
$$\neg p9)$$

$$p1 = T, p2 = F, p3 = F \; p4 = T$$
$$p5 = F, p6 = T, p7 = F, p8 = F,$$
$$p9 = F$$

$$x = 1, x \neq 3, y \neq 1 \; y = 2,$$
$$y \neq 3, z = 3, x \neq y, x \neq z,$$
$$y \neq z$$

Lazy SMT Techniques



Architecture of Z3's SMT Core solver

EUF

The logic of **equality and uninterpreted function**, EUF, is a basic ingredient for (first-order) predicate logic.

While formulas are not empty:

remove a=b;

merge(a, b);

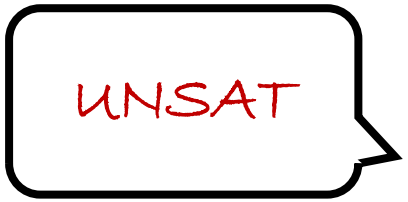
If find(a) = find(b) for some a≠b

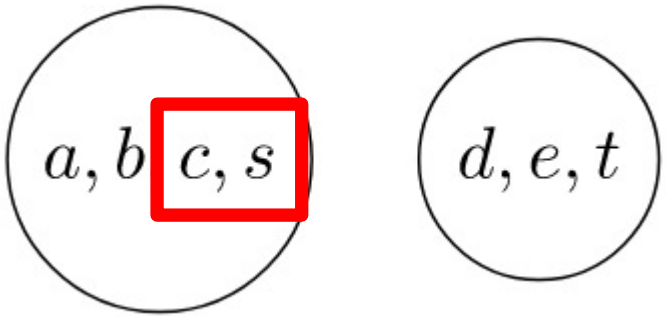
return false;

else

return true;

$$a = b, b = c, d = e$$
$$b = s, d = t, c \neq s$$





EUF

THEOREM

Congruence rule:

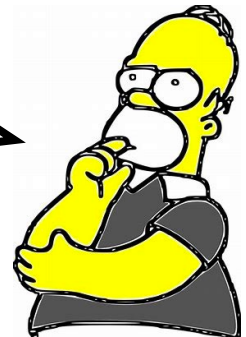
$$x_1 = y_1, \ldots, x_n = y_n \Rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$$

$$a = b, b = c, d = e$$

$$b = s, d = t$$

$$f(a, g(d)) \neq f(b, g(e))$$

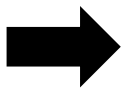
When there are functions, we can construct the graph with congruence rule.



EUF

1) introduce constants that can be used as shorthands for sub-terms.

$$a = b, b = c, d = e$$
$$b = s, d = t$$
$$f(a, g(d)) \neq f(b, g(e))$$



$$a = b, b = c, d = e$$
$$b = s, d = t, v3 \neq v4$$

$$v1 := g(e), v2 := g(d)$$
$$v3 := f(a, v2), v4 := f(b, v1)$$

EUF

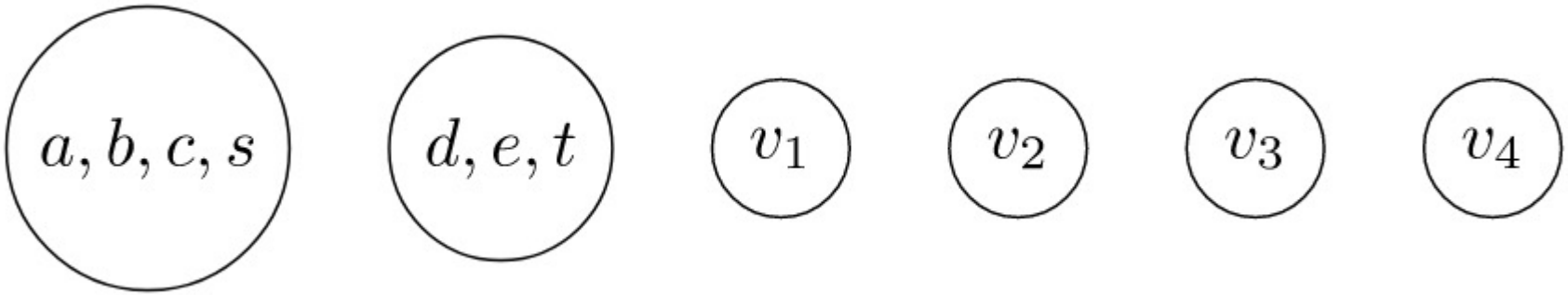
2) Working bottom-up, the congruence rule dictates how to merge groups

$$a = b, b = c, d = e$$

$$b = s, d = t, v3 \neq v4$$

$$v1 := g(e), v2 := g(d)$$

$$v3 := f(a, v2), v4 := f(b, v1)$$

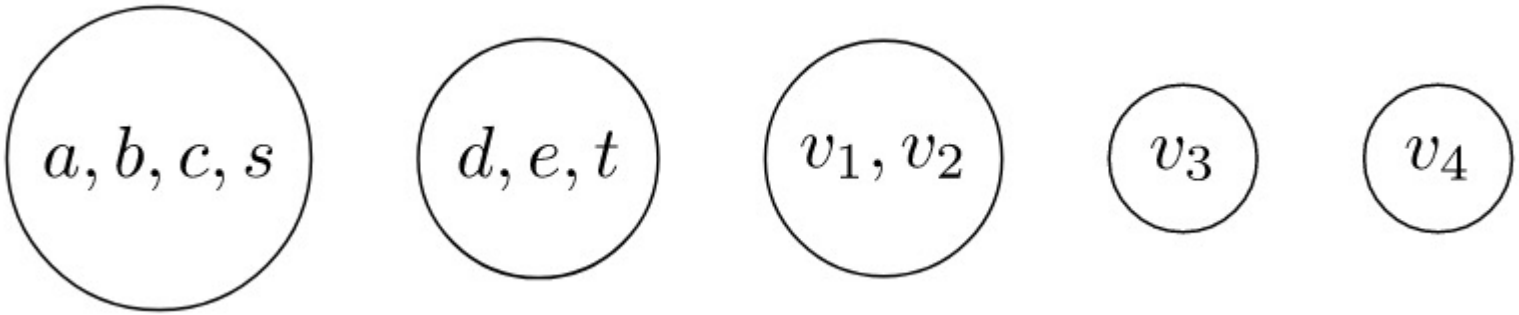


EUF

2) Working bottom-up, the congruence rule dictates how to merge groups

$$a = b, b = c, d = e$$
$$b = s, d = t, v3 \neq v4$$
$$v1 := g(e), v2 := g(d)$$
$$v3 := f(a, v2), v4 := f(b, v1)$$

$e = d \Rightarrow g(e) = g(d)$

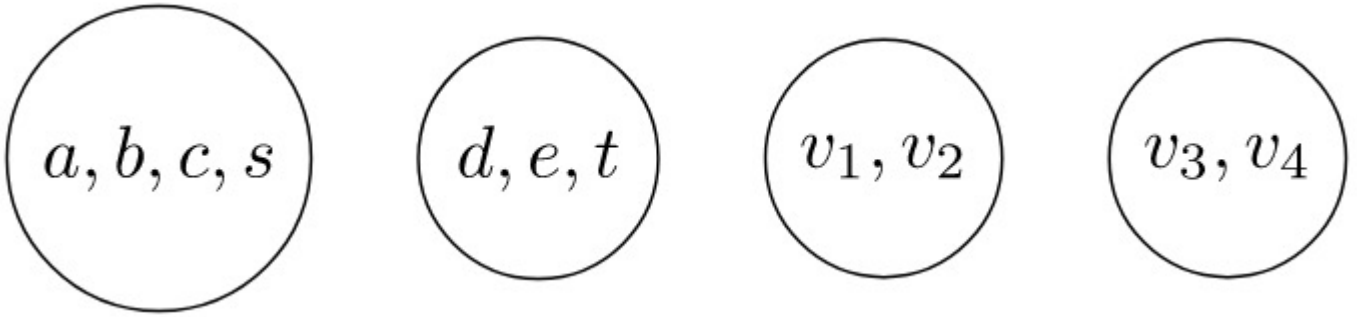


EUF

2) Working bottom-up, the congruence rule dictates how to merge groups

$$a = b, v2 = v1 \Rightarrow f(a, v2) = f(b, v1)$$

$$a = b, b = c, d = e$$
$$b = s, d = t, v3 \neq v4$$
$$v1 := g(e), v2 := g(d)$$
$$v3 := f(a, v2), v4 := f(b, v1)$$

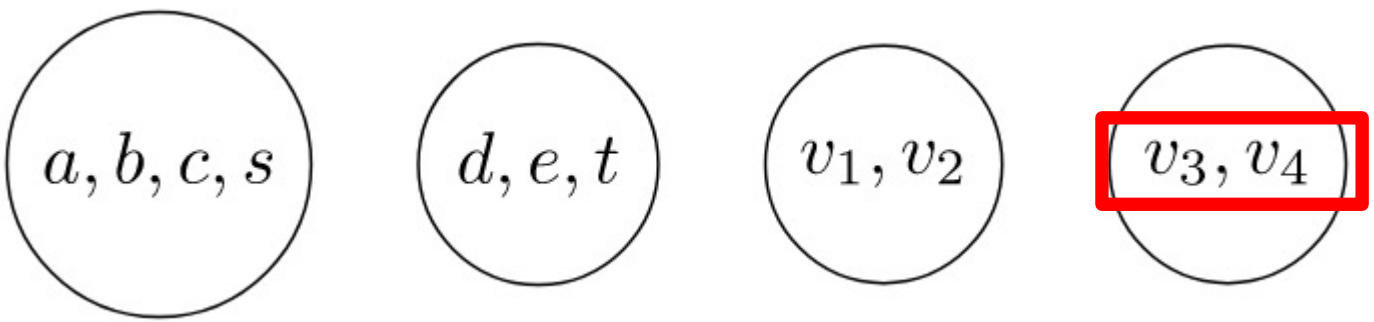


EUF

3) Check satisfiability like before

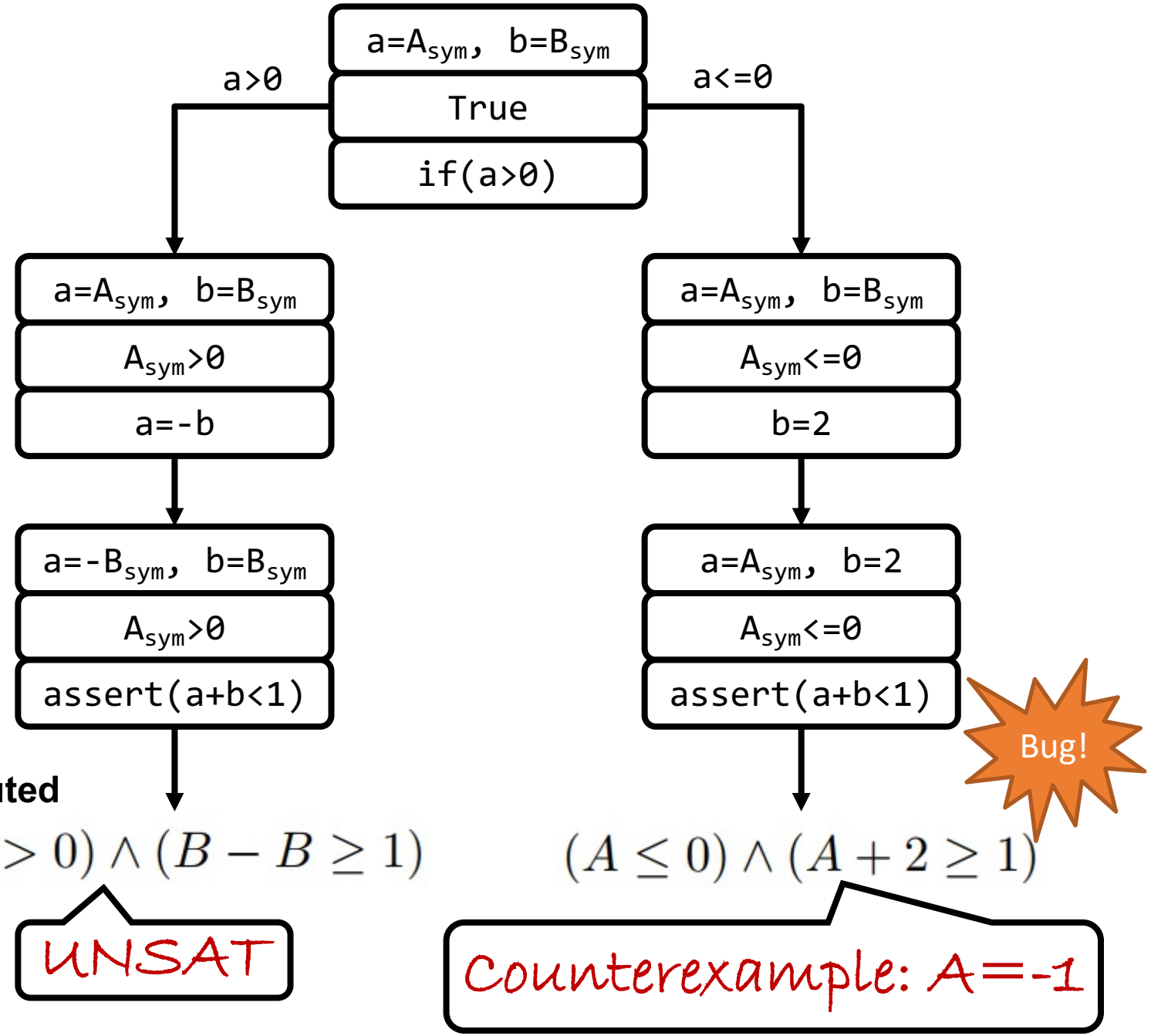
$$a = b, b = c, d = e$$
$$b = s, d = t, v3 \neq v4$$
$$v1 := g(e), v2 := g(d)$$
$$v3 := f(a, v2), v4 := f(b, v1)$$

UNSAT



Symbolic Execution

```
void func(int a, int b) {
  if(a > 0)
    a = -b;
  else
    b = 2;
  assert(a + b < 1);
}
```



Path constraints: the path condition

Next code: the next instruction to be executed

Symbolic store: the symbolic value for each variable

Part I: Computer Aided Reasoning Overview

1.1 Propositional Logic

Step 1. Convert program into propositional formula.

Step 2. Ask the computer to solve the formula.

1.2 First Order Logic

Step 1. Convert program 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

Previous verification method cannot address unbounded loop. We introduce Hoare logic.

Defining Correctness

DEFINITION

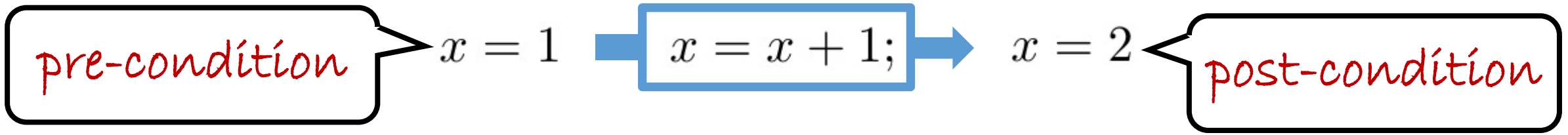
Program state includes the value of every variable used in the program.

DEFINITION

Pre-condition describes the program state before executing the program.

DEFINITION

Post-condition describes the program state after executing the program.

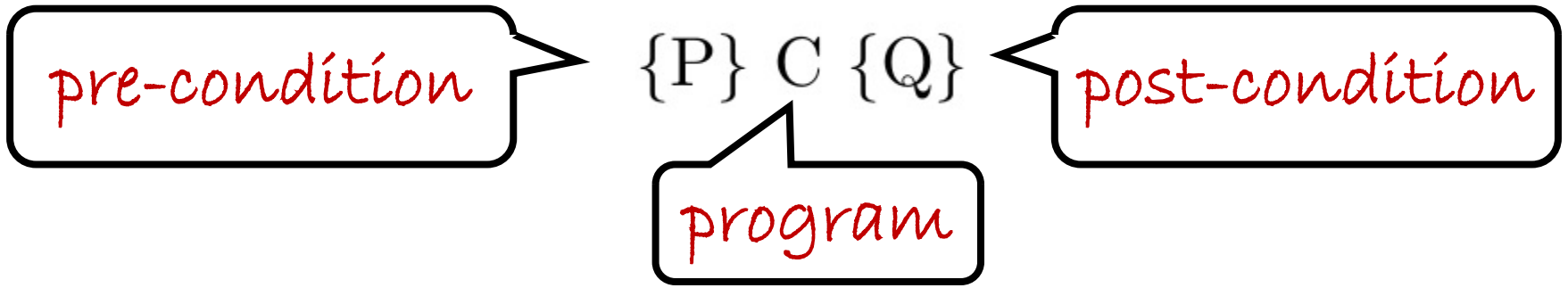


Hoare Triple (霍尔三元组)

DEFINITION

A program C is partially correct iff:

whenever C is executed in a state initially satisfying pre-condition and if the execution of C terminates, then the state in which C's execution terminates satisfies post-condition.



Proving Correctness

Replace a in P with v

$$\{P\}a := e\{(\exists v)(a = e[v/a] \wedge P[v/a])\}$$

$$\frac{P \rightarrow Q \quad \{Q\}C\{R\}}{\{P\}C\{R\}}$$

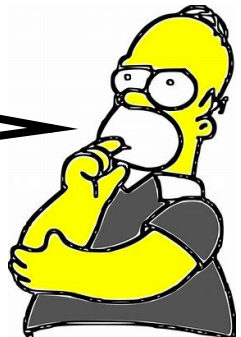
$$\{P[e/a]\}a := e\{P\}$$

$$\frac{\{P\}C\{Q\} \quad Q \rightarrow R}{\{P\}C\{R\}}$$

$$\frac{\{P\}C1\{R\} \quad \{R\}C2\{Q\}}{\{P\}C1;C2\{Q\}}$$

$$\frac{\{P \wedge istrue(b1)\}C1\{Q\} \quad \{P \wedge isfalse(b1)\}C2\{Q\}}{\{P\}if(b1)\ then\ C1\ else\ C2\ \{Q\}}$$

A Hoare triple can be proved by these axioms and inference rules.



Loop Invariant

loop invariant

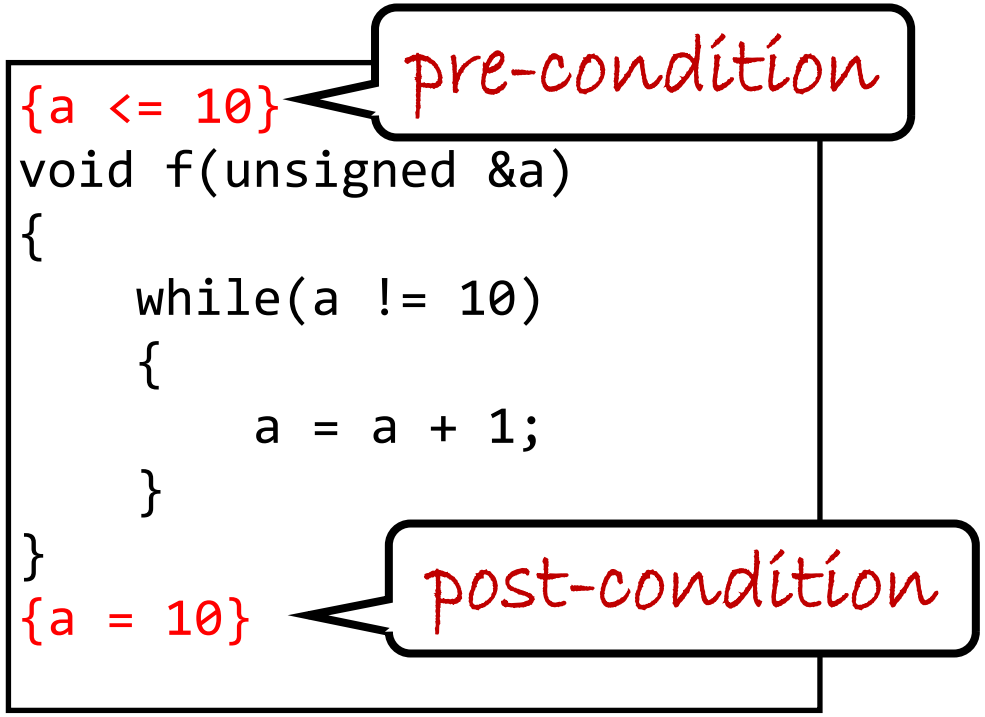
$$\frac{\{I \wedge istrue(b1)\}C\{I\}}{\{I\}\ while(b1)\ C\ \{I \wedge isfalse(b1)\}}$$

The while rule (WHP)

It says that

- if executing C once preserves the truth of I, then executing C any number of times also preserves the truth of I
- after a while loop has terminated, the test must be false

Loop Invariant



I: a <= 10

loop invariant

Loop Invariant

$$\{I \wedge istrue(b1)\}C\{I\}$$

$$\{I\}\ while(b1)\ C\ \{I \wedge isfalse(b1)\}$$

$\{a \leq 10\}\ while(a! = 10)\ a := a + 1\ \{a = 10\}$

Proof.

(1) $\{a + 1 \leq 10\}a := a + 1\{a \leq 10\}$ — AS

(2) $a \leq 10 \wedge a \neq 10 \rightarrow a + 1 \leq 10$ — *predicate logic*

(3) $\{a \leq 10 \wedge a \neq 10\}a := a + 1\{a \leq 10\}$ — $SP(1)(2)$

(4) $\{a \leq 10\}\ while(a! = 10)\ a := a + 1\ \{a \leq 10 \wedge a = 10\}$ — $WHP(3)$

(5) $a \leq 10 \wedge a = 10 \rightarrow a = 10$ — *predicate logic*

(6) $\{a \leq 10\}\ while(a! = 10)\ a := a + 1\ \{a = 10\}$ — $WC(4)(5)$

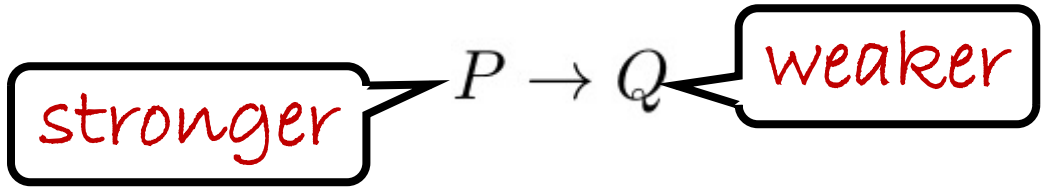
assignment

强化前置条件

While loop rule

弱化后置条件

Weakest Precondition



DEFINITION

Given a program C and postcondition Q, P is the weakest precondition means for all P', {P'} C {Q} iff P' → P. We use wp(C, Q) to denote P.

$$\{a = 2\}a := a + 1\{a = 3\}$$

$$\{a = 2 \wedge b = 4\}a := a + 1\{a = 3\}$$

$$\{a = 2 \wedge a > 1\}a := a + 1\{a = 3\}$$

weakest precondition

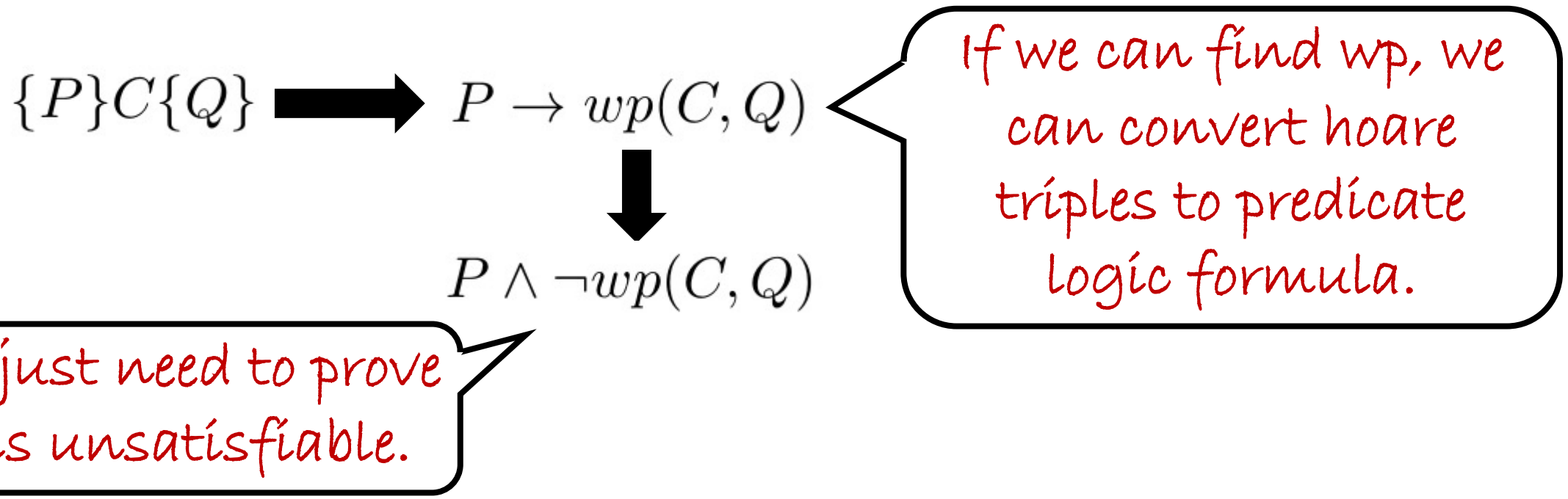
$$a = 2 \wedge b = 4 \rightarrow a = 2$$

$$a = 2 \wedge a > 1 \rightarrow a = 2$$

Basic Idea of Automated Verification

To prove {P}C{Q}

- Find the wp(C, Q), then prove P implies wp(C, Q).

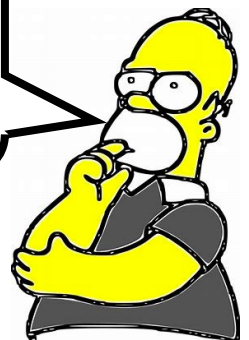


Weakest Liberal Precondition

We use wlp(C, Q) instead of wp(C, Q) in following slides:

- C is the program, Q is the intended postcondition
- Name stands for weakest “liberal” precondition: it ignores termination
- For all C and Q, it is guaranteed that { wlp(C, Q) } C {Q} is true
- For all P, C and Q, {P} C {Q} iff $P \rightarrow wlp(C, Q)$

Wlp must be computable. We will introduce how to compute it automatically.



Weakest Liberal Precondition

$$wlp(skip, Q) = Q$$

$$wlp(a := e, P) = P[e/a]$$

$$wlp(C1; C2, Q) = wlp(C1, wlp(C2, Q))$$

$$wlp(if(b1)\ then\ \{C1\}\ else\ \{C2\}, Q)$$

$$=(istrue(b1) \rightarrow wlp(C1, Q)) \wedge (isfalse(b1) \rightarrow wlp(C2, Q))$$

While Loop

$$\frac{\{I \wedge istrue(b1)\}C\{I\}}{\{I\}\ while(b1)\ C\ \{I \wedge isfalse(b1)\}}$$

$$\{P\}while(b1)C\{Q\} \quad \begin{cases} P \rightarrow I \\ (I \wedge isfalse(b1)) \rightarrow Q \\ \{I \wedge istrue(b1)\}C\{I\} \end{cases}$$

Invariant need be written manually

Thanks & Questions