Satisfiability Modulo Theory (SMT)

Testing, Quality Assurance, and Maintenance
Winter 2019

Prof. Arie Gurfinkel

UNIVERSITY OF WATERLOO

Satisfiability Modulo Theory (SMT)

Satisfiability is the problem of determining wither a formula F has a model

- if F is ***propositional***, a model is a truth assignment to Boolean variables
- if F is ***first-order formula***, a model assigns values to variables and interpretation to all the function and predicate symbols

SAT Solvers

- check satisfiability of propositional formulas

SMT Solvers

- check satisfiability of formulas in a ***decidable*** first-order theory (e.g., linear arithmetic, uninterpreted functions, array theory, bit-vectors)

Background Reading: SMT

September 2011

isfiability Modulo Theories: Introduction & Applications

Leonardo de Moura
Microsoft Research
One Microsoft Way
Redmond, WA 98052
leonardo@microsoft.com

Nikolaj Bjørner
Microsoft Research
One Microsoft Way
Redmond, WA 98052
nbjorner@microsoft.com

RACT

aint satisfaction problems arise in many diverse ar- uding software and hardware verification, type infer- atic program analysis, test-case generation, schedul- anning and graph problems. These areas share a n trait, they include a core component using logical s for describing states and transformations between The most well-known constraint satisfaction problem ositional satisfiability, SAT, where the goal is to de- ether a formula over Boolean variables, formed using connectives can be made *true* by choosing *true*/*false* for its variables. Some problems are more naturally ed using richer languages, such as arithmetic. A sup- *theory* (of arithmetic) is then required to capture ning of these formulas. Solvers for such formulations mmonly called *Satisfiability Modulo Theories* (SMT)

SMT solvers have been the focus of increased recent attention thanks to technological advances and industrial applications. Yet, they draw on a combination of some of the most fundamental areas in computer science as well as discoveries from the past century of symbolic logic. They combine the problem of Boolean Satisfiability with domains, such as, those studied in convex optimization and term-manipulating symbolic systems. They involve the decision problem, completeness and incompleteness of logical theories, and finally complexity theory. In this article, we present an overview of the field of Satisfiability Modulo Theories, and some of its applications.

key driving factor [4]. An important ingredient is a common interchange format for benchmarks, called SMT-LIB [33], and the classification of benchmarks into various categories depending on which theories are required. Conversely, a growing number of applications are able to generate benchmarks in the SMT-LIB format to further inspire improving SMT solvers.

There is a relatively long tradition of using SMT solvers in select and specialized contexts. One prolific case is theorem proving systems such as ACL2 [26] and PVS [32]. These use decision procedures to discharge lemmas encountered during interactive proofs. SMT solvers have also been used for a long time in the context of program verification and *extended static checking* [21], where verification is focused on assertion checking. Recent progress in SMT solvers, however, has enabled their use in a set of diverse applications, including interactive theorem provers and extended static checkers, but also in the context of scheduling, planning, test-case generation, model-based testing and program development, static program analysis, program synthesis, and run-time analysis, among several others.

We begin by introducing a motivating application and a simple instance of it that we will use as a running example.

1.1 An SMT Application - Scheduling

Consider the classical *job shop scheduling* decision problem. In this problem, there are n jobs, each composed of m tasks of varying duration that have to be performed consecutively on m machines. The start of a new task can be delayed as long as needed in order to wait for a machine to become available, but tasks cannot be interrupted once

UNIVERSITY OF WATERLOO

Example

$$b + 2 = c \wedge f(\texttt{read}(\texttt{write}(a, b, 3), c - 2)) \neq f(c - b + 1)$$

Example

$$b + 2 = c \wedge f(\texttt{read}(\texttt{write}(a, b, 3), c - 2)) \neq f(c - b + 1)$$

Arithmetic

Example

$$b + 2 = c \wedge f(\texttt{read}(\texttt{write}(a, b, 3), c - 2)) \neq f(c - b + 1)$$

Array theory

Example

$$b + 2 = c \wedge f(\texttt{read}(\texttt{write}(a, b, 3), c - 2)) \neq f(c - b + 1)$$

Uninterpreted function

Example

$$b + 2 = c \wedge f(\texttt{read}(\texttt{write}(a, b, 3), c - 2)) \neq f(c - b + 1)$$

Example

$$b + 2 = c \land f(\mathtt{read}(\mathtt{write}(a, b, 3), c - 2)) \neq f(c - b + 1)$$

By arithmetic, this is equivalent to

$$b + 2 = c \land f(\mathtt{read}(\mathtt{write}(a, b, 3), b)) \neq f(3)$$

Example

$$b + 2 = c \wedge f(\texttt{read}(\texttt{write}(a, b, 3), c - 2)) \neq f(c - b + 1)$$

By arithmetic, this is equivalent to

$$b + 2 = c \wedge f(\texttt{read}(\texttt{write}(a, b, 3), b)) \neq f(3)$$

then, by the array theory axiom: $\texttt{read}(\texttt{write}(v, i, x), i) = x$

$$b + 2 = c \wedge f(3) \neq f(3)$$

Example

$$b + 2 = c \land f(\texttt{read}(\texttt{write}(a, b, 3), c - 2)) \neq f(c - b + 1)$$

By arithmetic, this is equivalent to

$$b + 2 = c \land f(\texttt{read}(\texttt{write}(a, b, 3), b)) \neq f(3)$$

then, by the array theory axiom: $\texttt{read}(\texttt{write}(v, i, x), i) = x$

$$b + 2 = c \land f(3) \neq f(3)$$

then, the formula is unsatisfiable

Example 2

$$x \geq 0 \land f(x) \geq 0 \land y \geq 0 \land f(y) \geq 0 \land x \neq y$$

Example 2

$$x \geq 0 \land f(x) \geq 0 \land y \geq 0 \land f(y) \geq 0 \land x \neq y$$

This formula is satisfiable

Example 2

$$x \geq 0 \land f(x) \geq 0 \land y \geq 0 \land f(y) \geq 0 \land x \neq y$$

This formula is satisfiable:

Example model:

$$x \to 1$$

$$y \to 2$$

$$f(1) \to 0$$

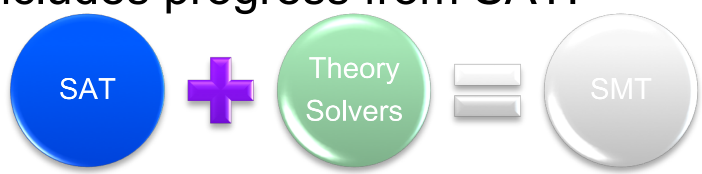
$$f(2) \to 1$$

$$f(\ldots) \to 0$$

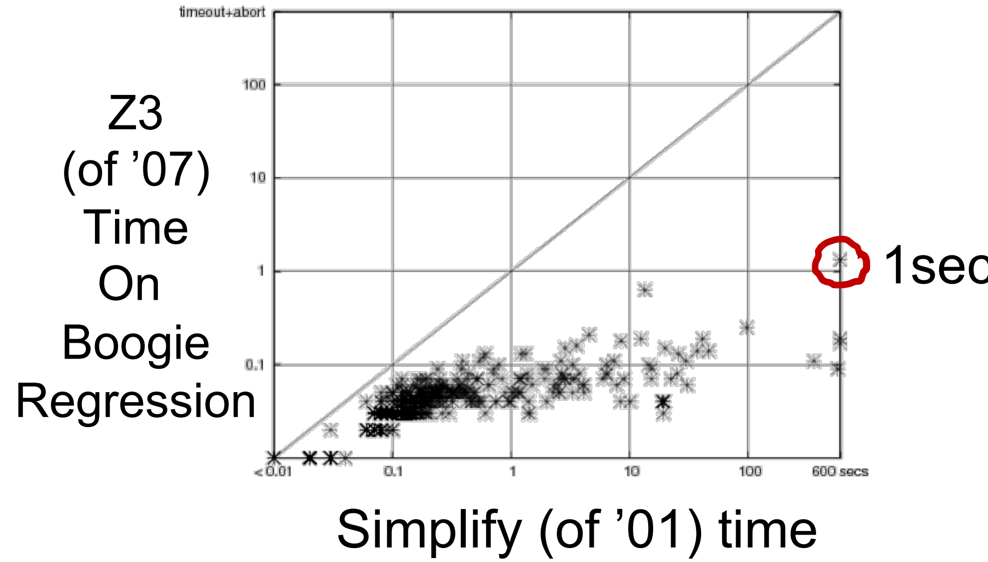
SMT - Milestones

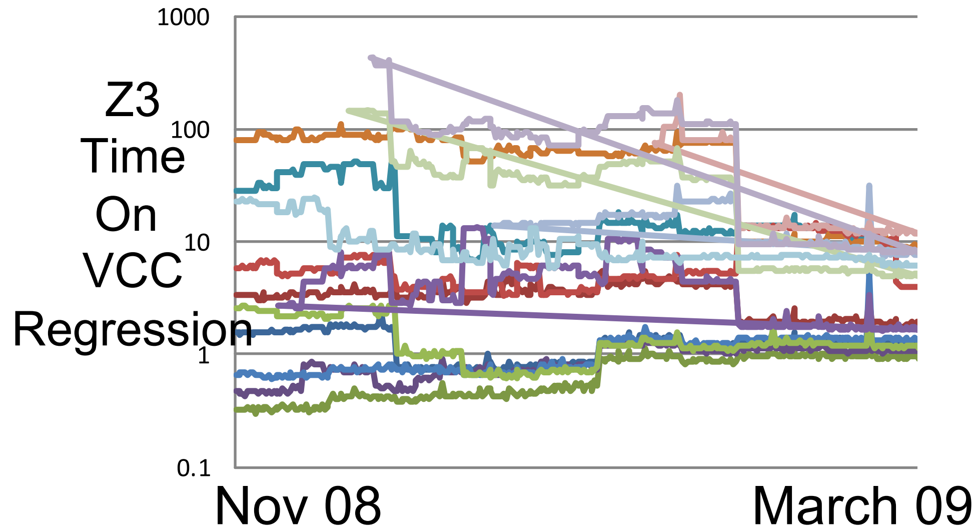
year	Milestone
1977	Efficient Equality Reasoning
1979	Theory Combination Foundations
1979	Arithmetic + Functions
1982	Combining Canonizing Solvers
1992-8	Systems: PVS, Simplify, STeP, SVC
2002	Theory Clause Learning
2005	SMT competition
2006	Efficient SAT + Simplex
2007	Efficient Equality Matching
2009	Combinatory Array Logic, …

Includes progress from SAT:



15KLOC + 285KLOC = Z3



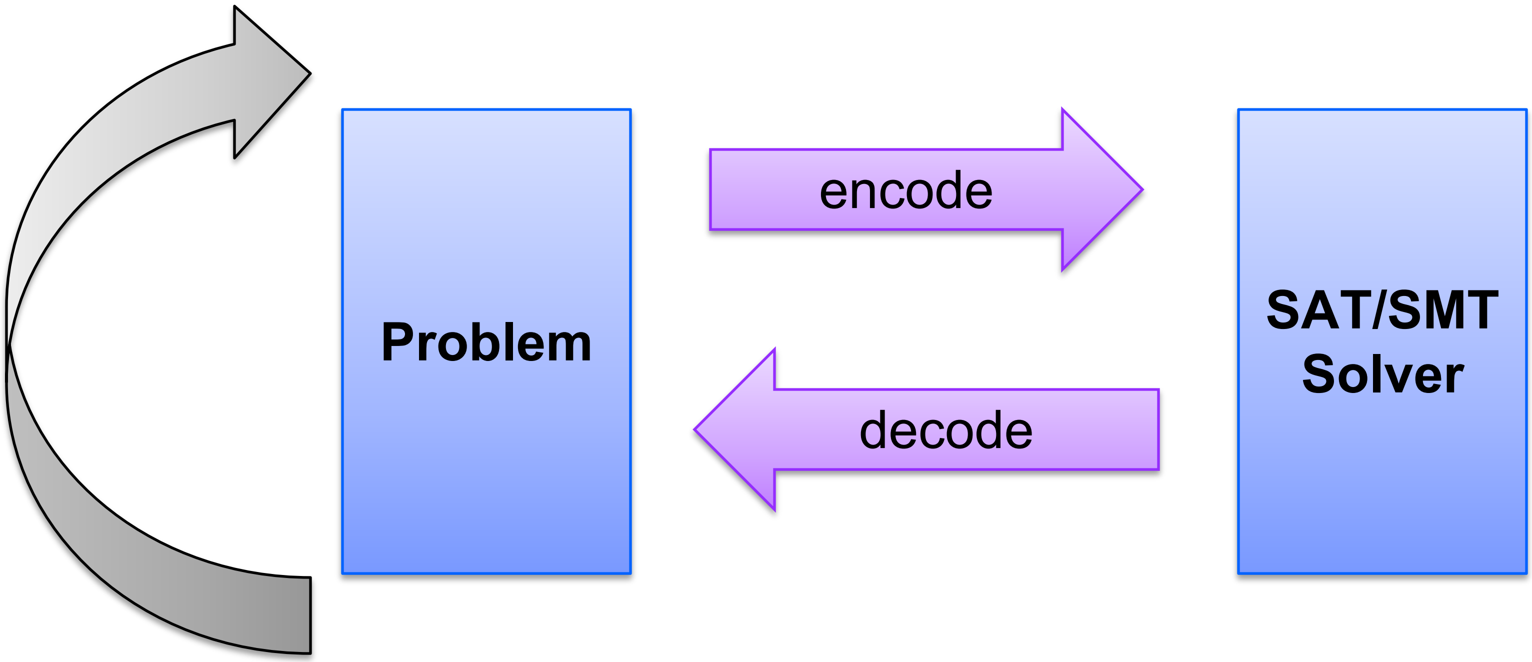


UNIVERSITY OF WATERLOO

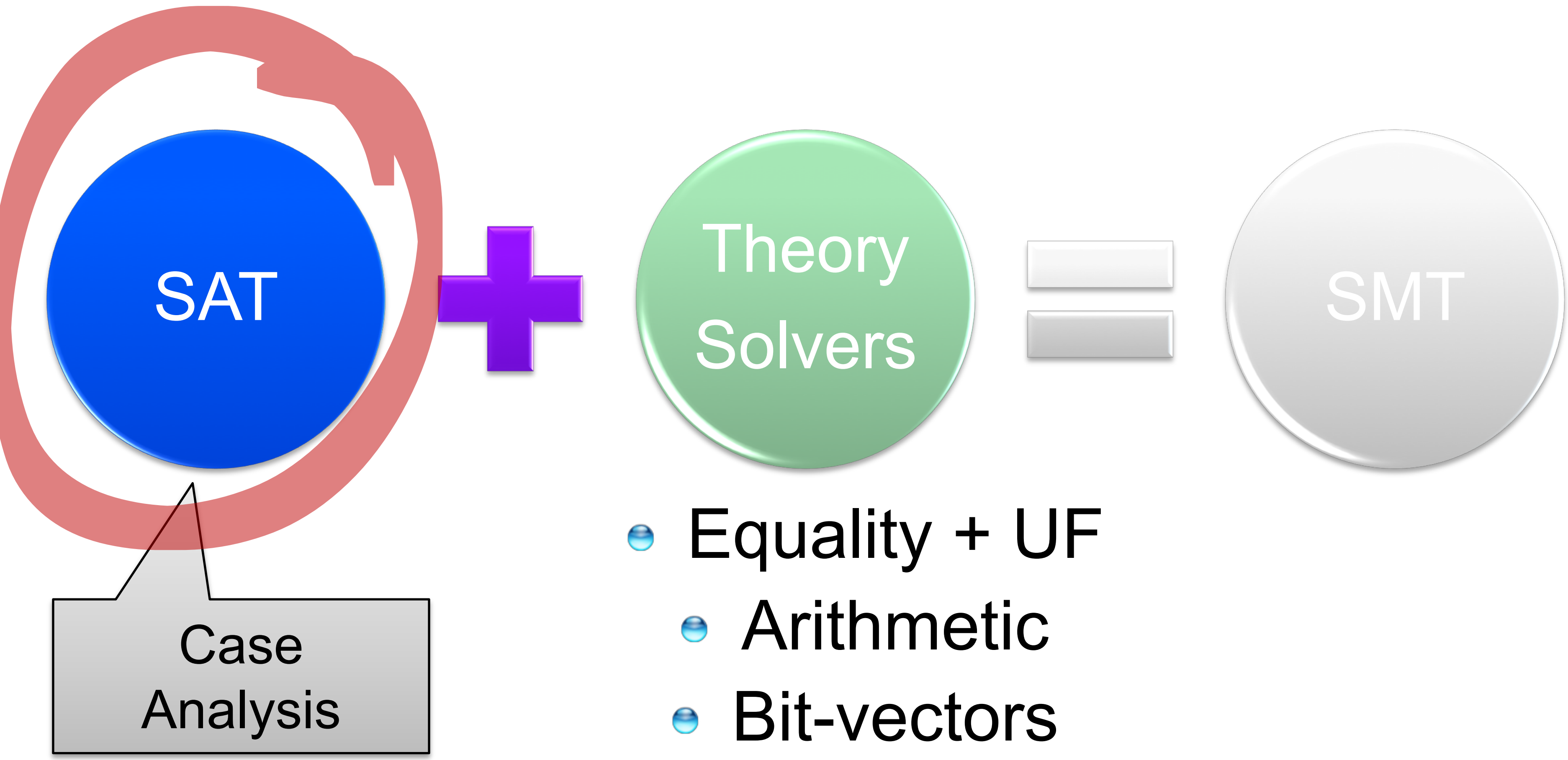
SAT/SMT Revolution

Solve any computational problem by effective reduction to SAT/SMT

- iterate as necessary



SMT : Basic Architecture



- Equality + UF
- Arithmetic
- Bit-vectors
- …

SAT + Theory solvers

Basic Idea

$x \geq 0,\ y = x + 1,\ (y > 2 \vee y < 1)$

Abstract (aka "naming" atoms)

$p_1,\ \ p_2,\ (p_3 \vee p_4)$ $\quad p_1 \equiv (x \geq 0),\ p_2 \equiv (y = x + 1),$
$p_3 \equiv (y > 2),\ p_4 \equiv (y < 1)$

SAT + Theory solvers

Basic Idea

$x \geq 0,\ y = x + 1,\ (y > 2 \vee y < 1)$

Abstract (aka "naming" atoms)

$p_1,\ p_2,\ (p_3 \vee p_4)$

$p_1 \equiv (x \geq 0),\ p_2 \equiv (y = x + 1),$
$p_3 \equiv (y > 2),\ p_4 \equiv (y < 1)$

SAT
Solver

SAT + Theory solvers

Basic Idea

$$x \geq 0,\ y = x + 1,\ (y > 2 \vee y < 1)$$

Abstract (aka "naming" atoms)

$p_1,\ p_2,\ (p_3 \vee p_4)$

$p_1 \equiv (x \geq 0),\ p_2 \equiv (y = x + 1),$
$p_3 \equiv (y > 2),\ p_4 \equiv (y < 1)$

SAT Solver

Assignment

$p_1,\ p_2,\ \neg p_3,\ p_4$

UNIVERSITY OF WATERLOO

SAT + Theory solvers

Basic Idea

$x \geq 0,\ y = x + 1,\ (y > 2 \vee y < 1)$

Abstract (aka "naming" atoms)

$p_1,\ p_2,\ (p_3 \vee p_4)$ $p_1 \equiv (x \geq 0),\ p_2 \equiv (y = x + 1),$
$p_3 \equiv (y > 2),\ p_4 \equiv (y < 1)$

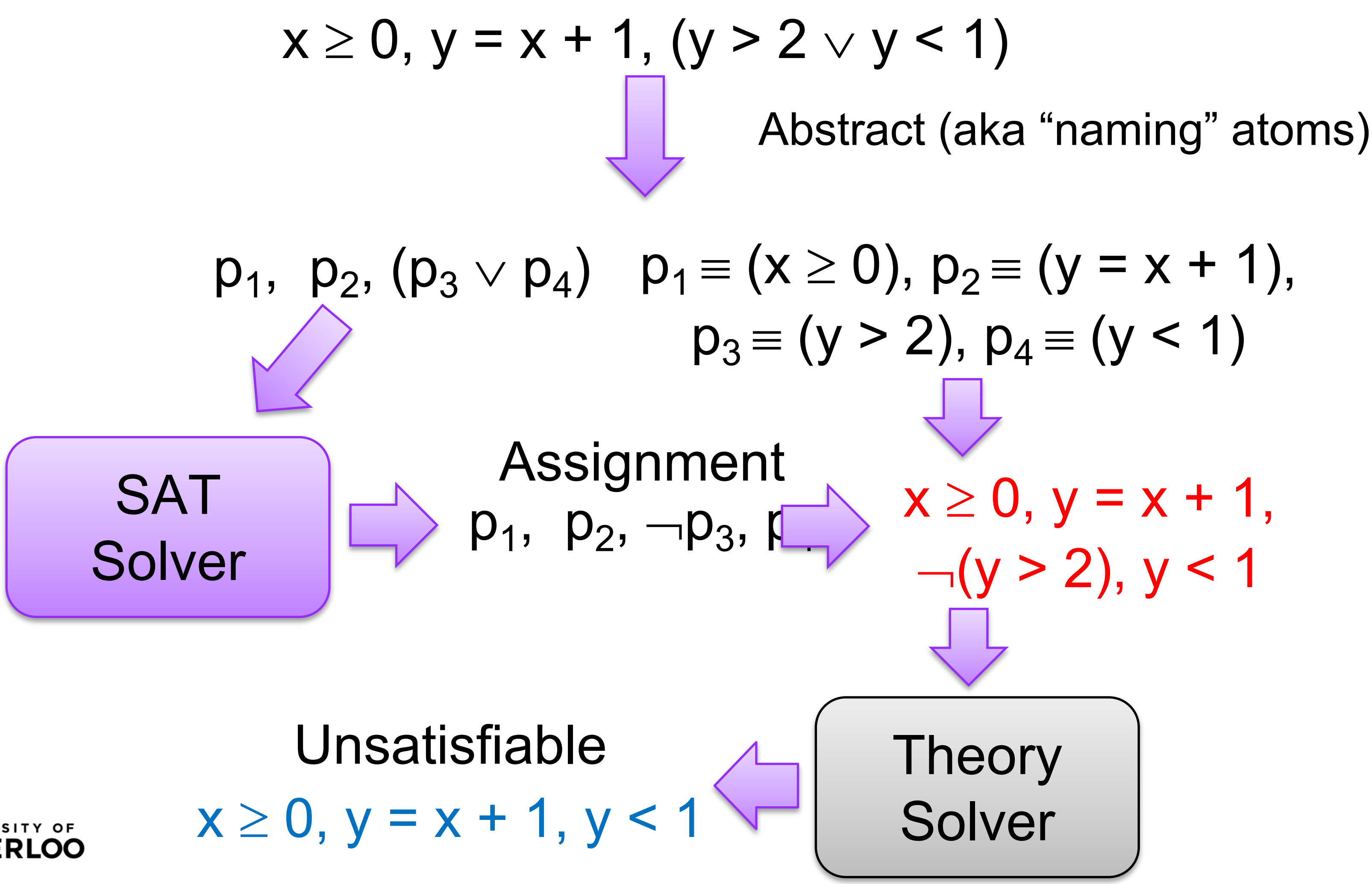
SAT Solver

Assignment
$p_1,\ p_2,\ \neg p_3,\ p$

$x \geq 0,\ y = x + 1,$
$\neg(y > 2),\ y < 1$

SAT + Theory solvers

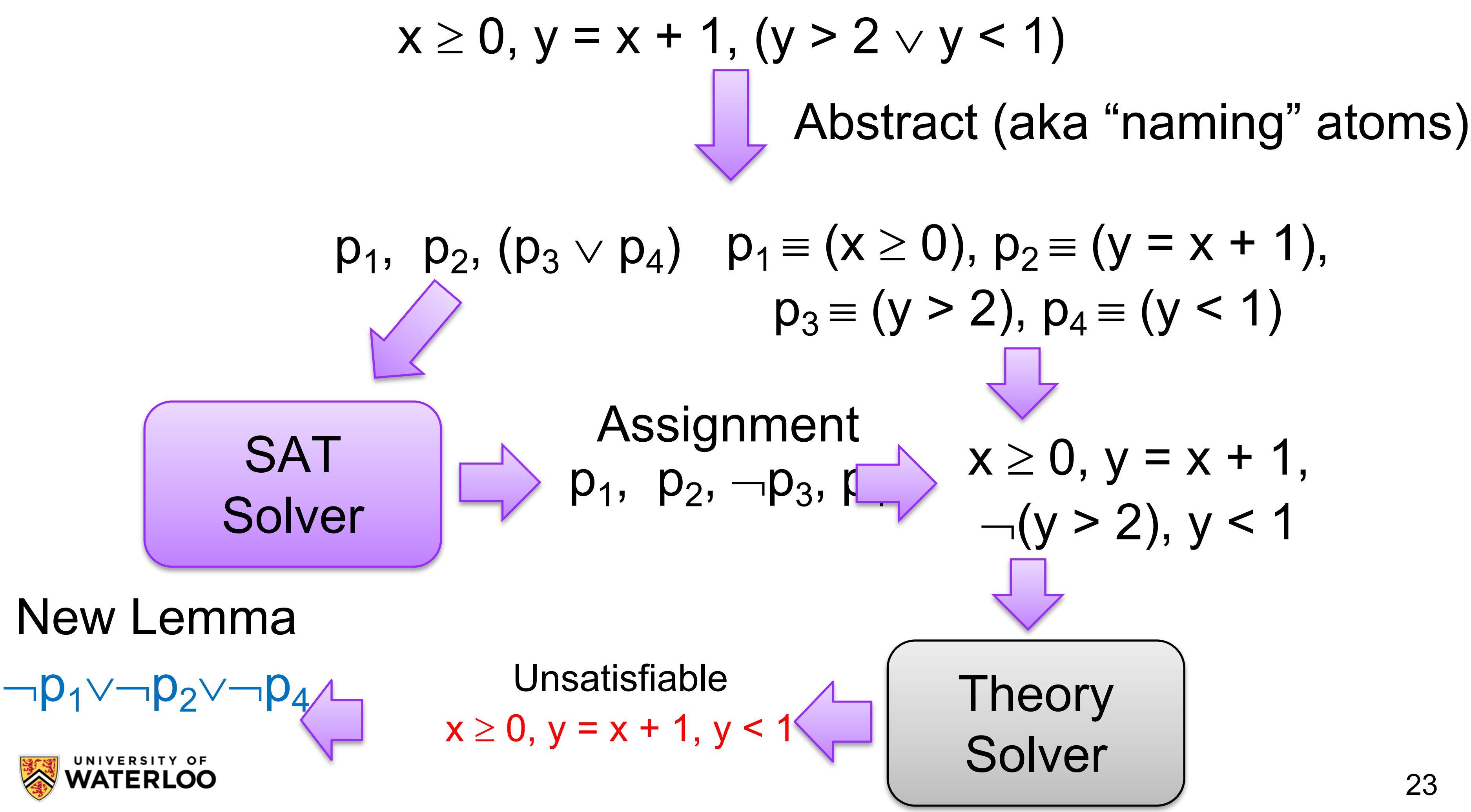
Basic Idea



UNIVERSITY OF WATERLOO

SAT + Theory solvers

Basic Idea



UNIVERSITY OF WATERLOO

SAT + Theory solvers

New Lemma ⇐ Unsatisfiable ⇐ Theory Solver

$\neg p_1 \vee \neg p_2 \vee \neg p_4$ ⇐ $x \geq 0, y = x + 1, y < 1$

AKA Theory conflict