Design of Theory Solvers in CVC4

Andrew Reynolds

8 April 2014

Satisfiability Modulo Theories

- SMT solvers used for :
  - Software/hardware verification
  - Automated Theorem Proving
  - Scheduling and Planning

SMT Solver : CVC4

- Joint project between NYU and U of Iowa
- State of the art successor of CVC3
- Based on DPLL(T) framework
- Supports wide range of theories

Theories supported by CVC4

- From SMT Lib :
  - Uninterpreted functions
  - Linear Integer and Real Arithmetic
  - Arrays
  - BitVectors
- Others :
  - Inductive Datatypes
  - Strings
  - Sets
  - Floating Points (coming soon)

SMT Example

$$x + 5 = \text{read}(A, 5) \wedge (A \neq B \vee \text{read}(B, 5) \leq f(x))$$

Arithmetic

Arrays

UF

SMT Example

$$x + 5 = \text{read}(A, 5) \wedge (A \neq B \vee \text{read}(B, 5) \leq f(x))$$

$\Downarrow$ Abstract to Propositional Logic

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

SMT Example

$$x + 5 = read(A, 5) \wedge (A \neq B \vee read(B, 5) \leq f(x))$$

$\Downarrow$ Abstract to Propositional Logic

$$\underbrace{P}_{\text{true}} \wedge (\underbrace{Q}_{\text{true}} \vee R)$$

SMT Example

$x + 5 = \text{read}(A, 5) \wedge (A \neq B \vee \text{read}(B, 5) \leq f(x))$

$\Downarrow$ Abstract to Propositional Logic

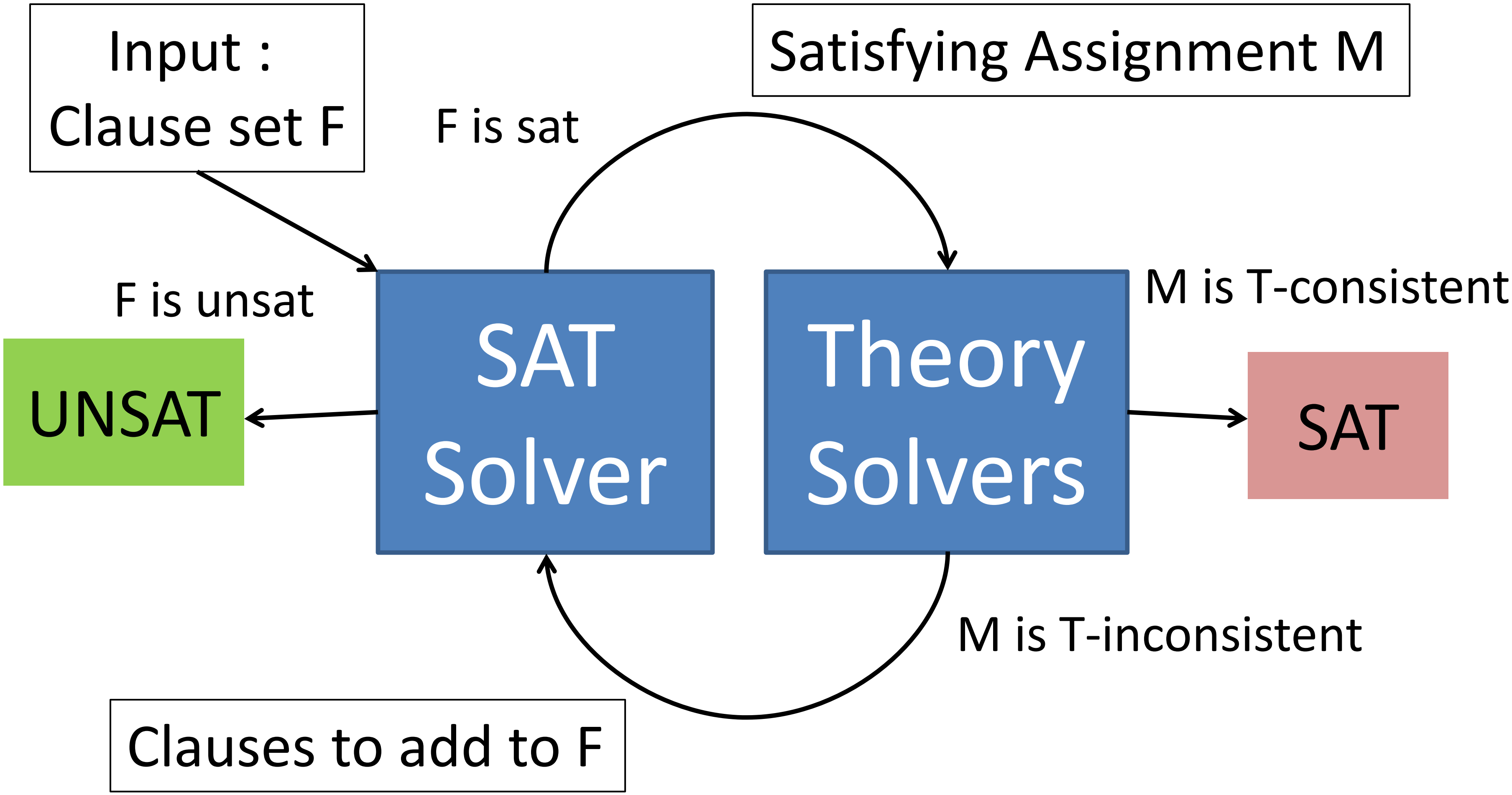
$P \wedge (Q \vee R)$

$\Downarrow$ Find satisfying assignment

$x + 5 = \text{read}(A, 5)$, $A \neq B$

$\Rightarrow$ Determine if *consistent* *according to theory*

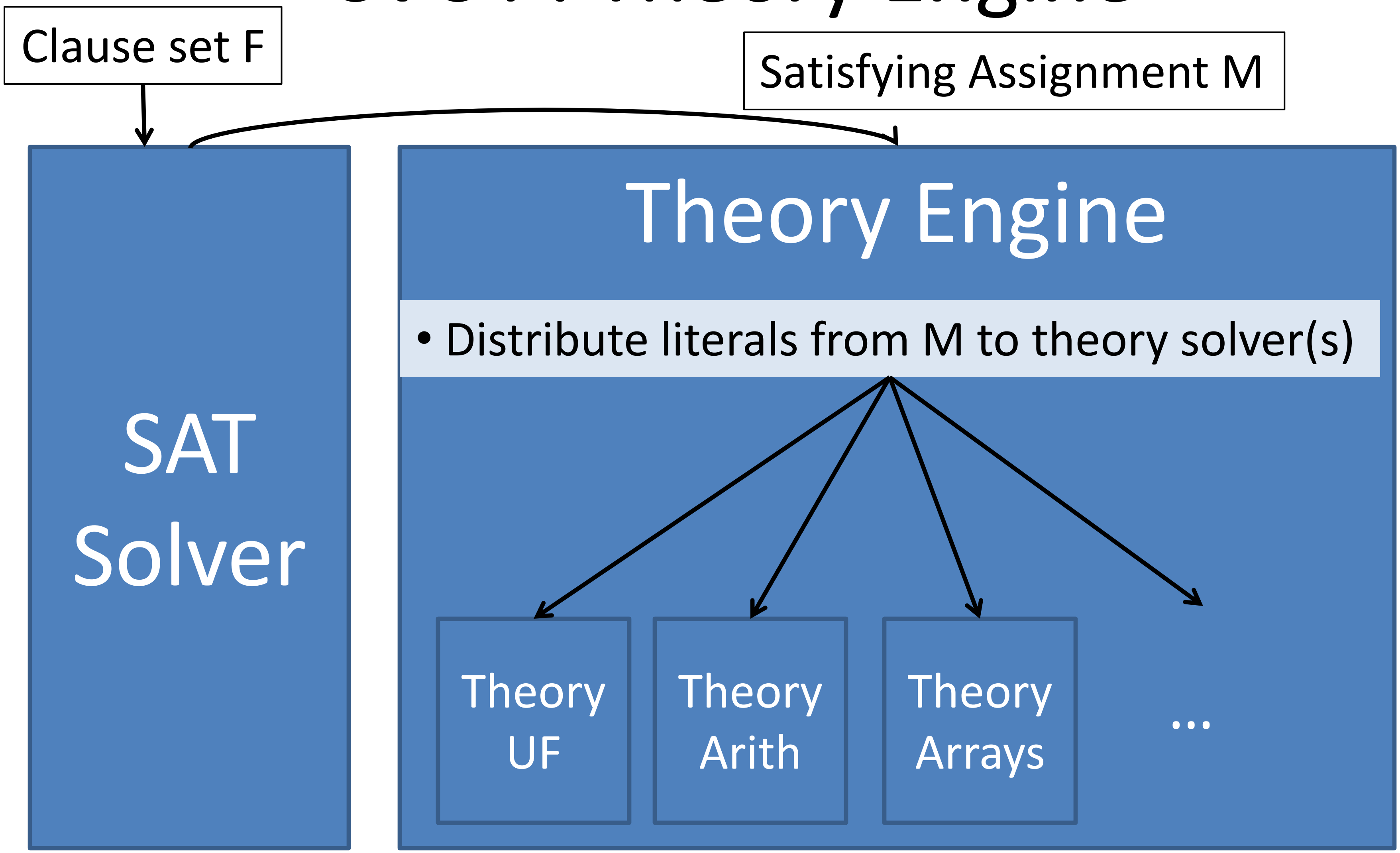
DPLL(T) Framework



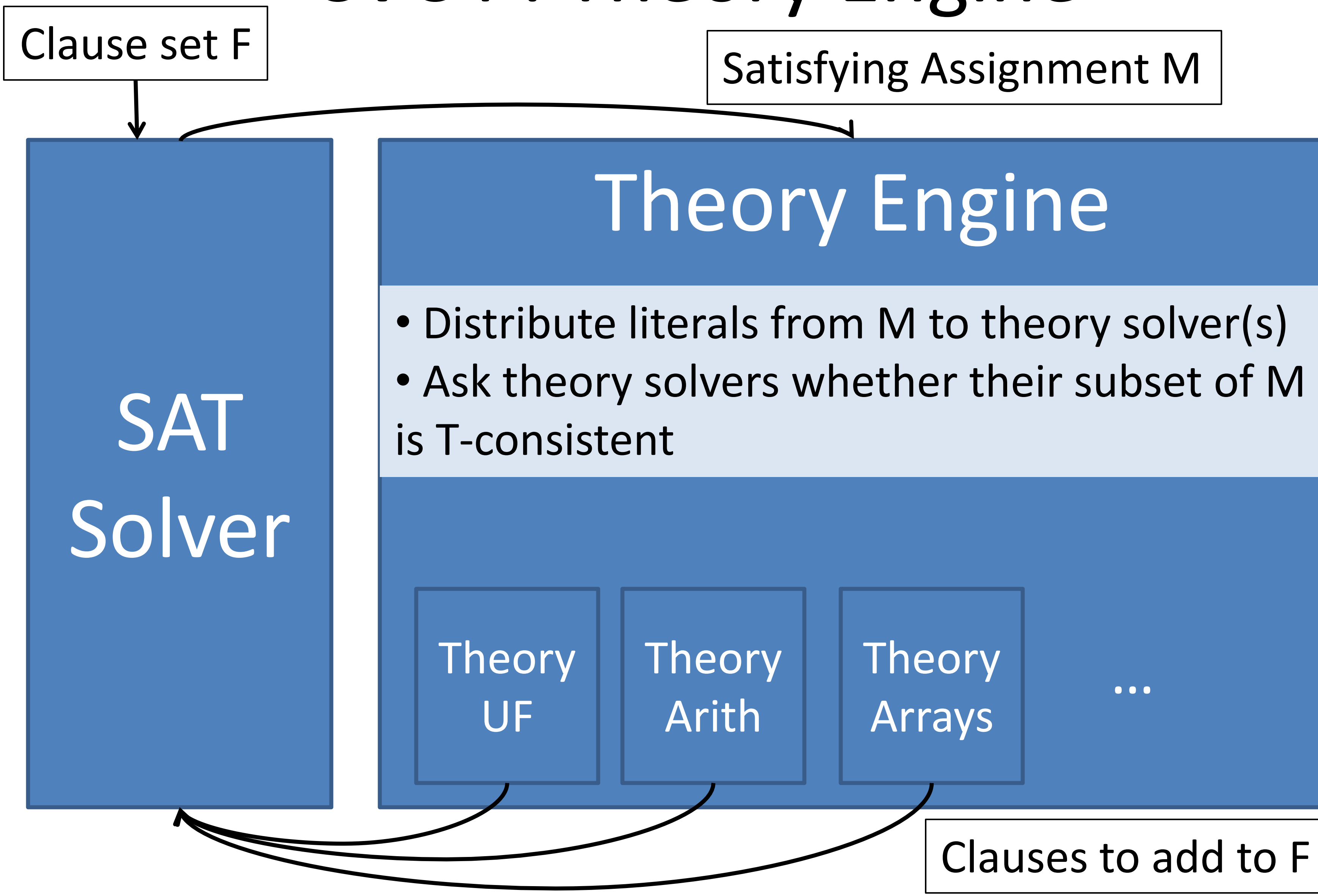
Architecture of CVC4

- CVC4 combines :
  - Off-the-shelf SAT solver (MiniSAT)
  - Multiple theory solvers
    - Managed by Theory Engine

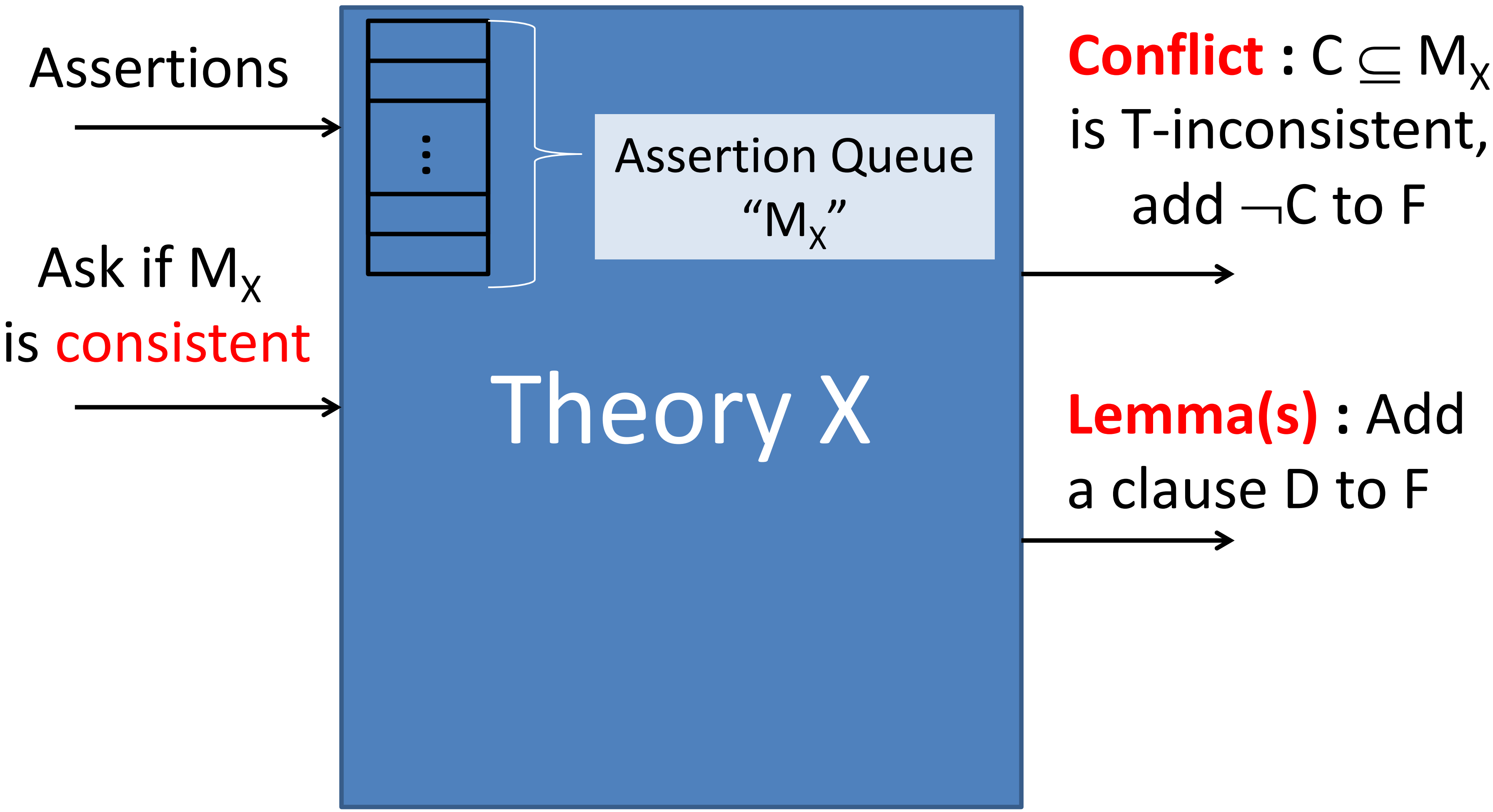
CVC4 : Theory Engine



CVC4 : Theory Engine



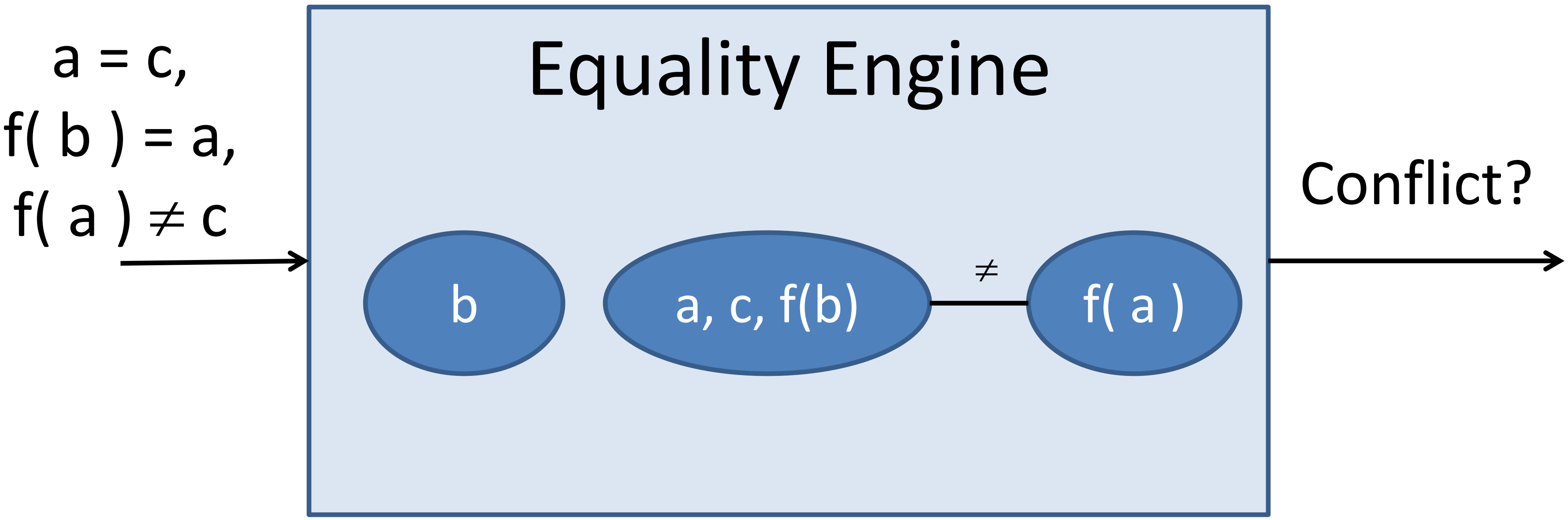
CVC4 : Theory Solvers



Handling Equality

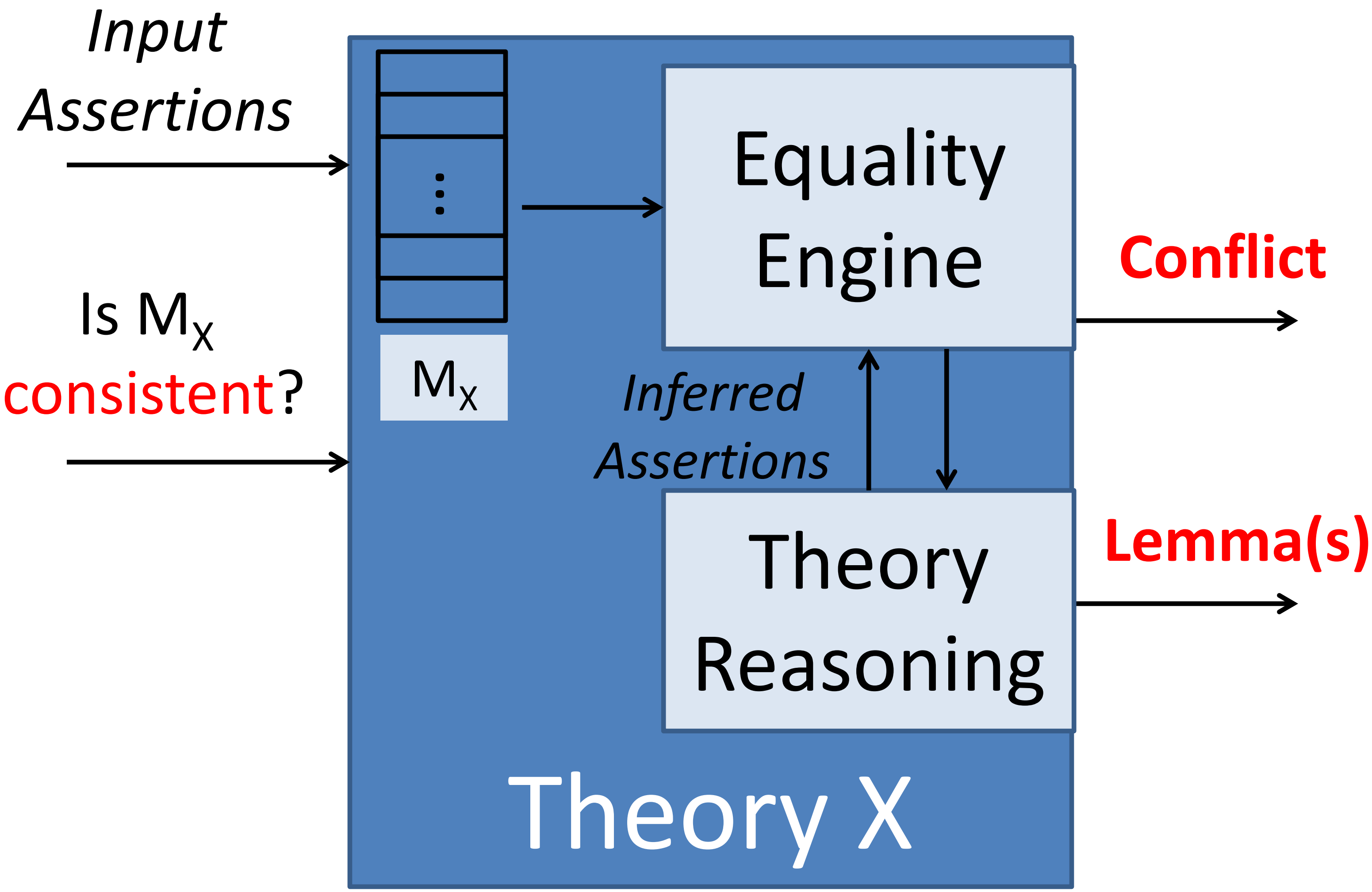
- Challenge : Equality reasoning is common to all theories, e.g.
  - $x + 1 = y$
  - $read(A, i) \neq read(A, j)$
  - $l_1 = cons(e, l_2)$
- *Idea* : Theory solvers use Equality Engine data structure

Equality Engine Data Structure



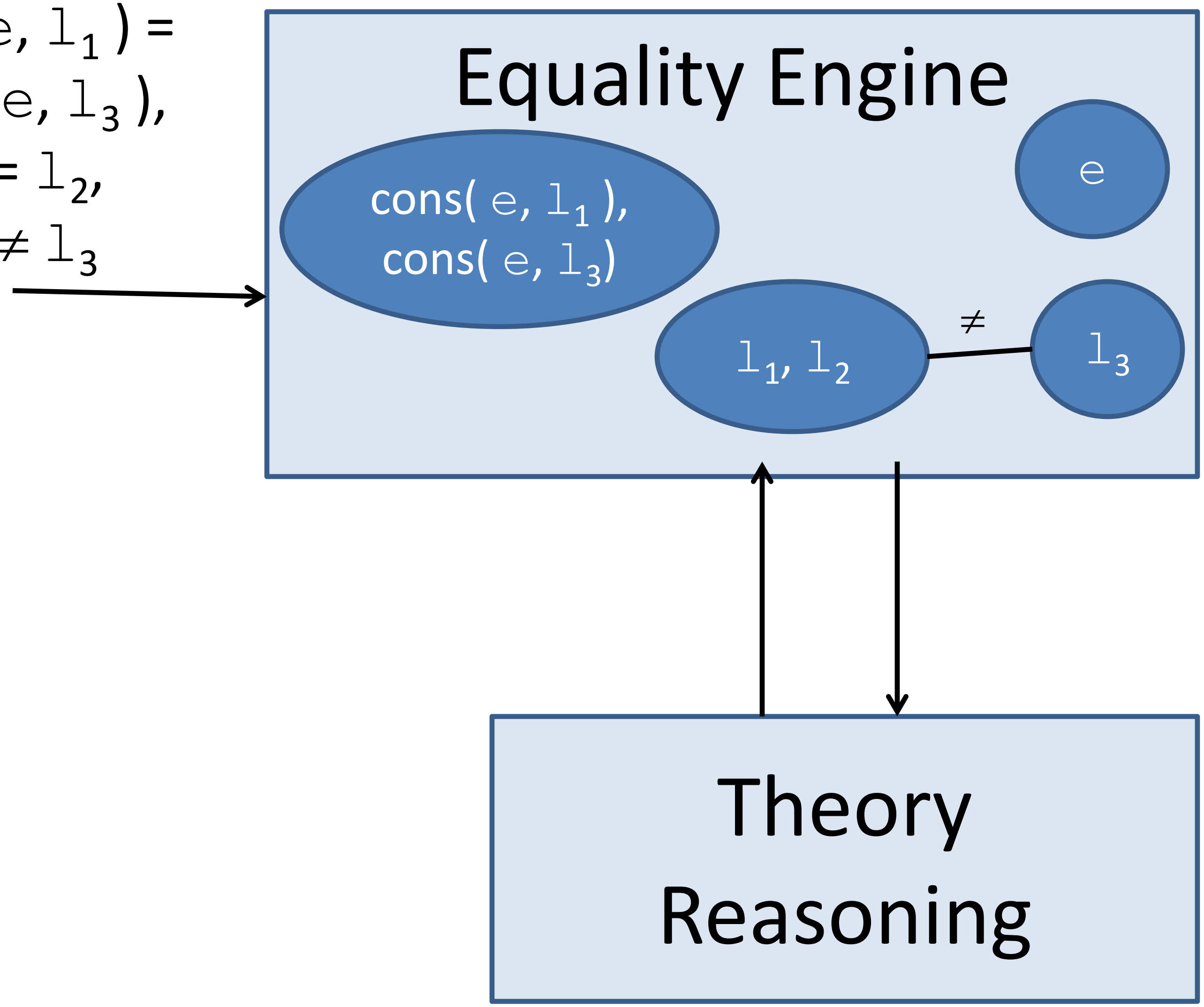
- Takes input a set of equalities and disequalities
  - Performs Congruence Closure
  - Maintains equivalence classes
  - Explains/reports conflicts

Theory Solver : Equality Engine

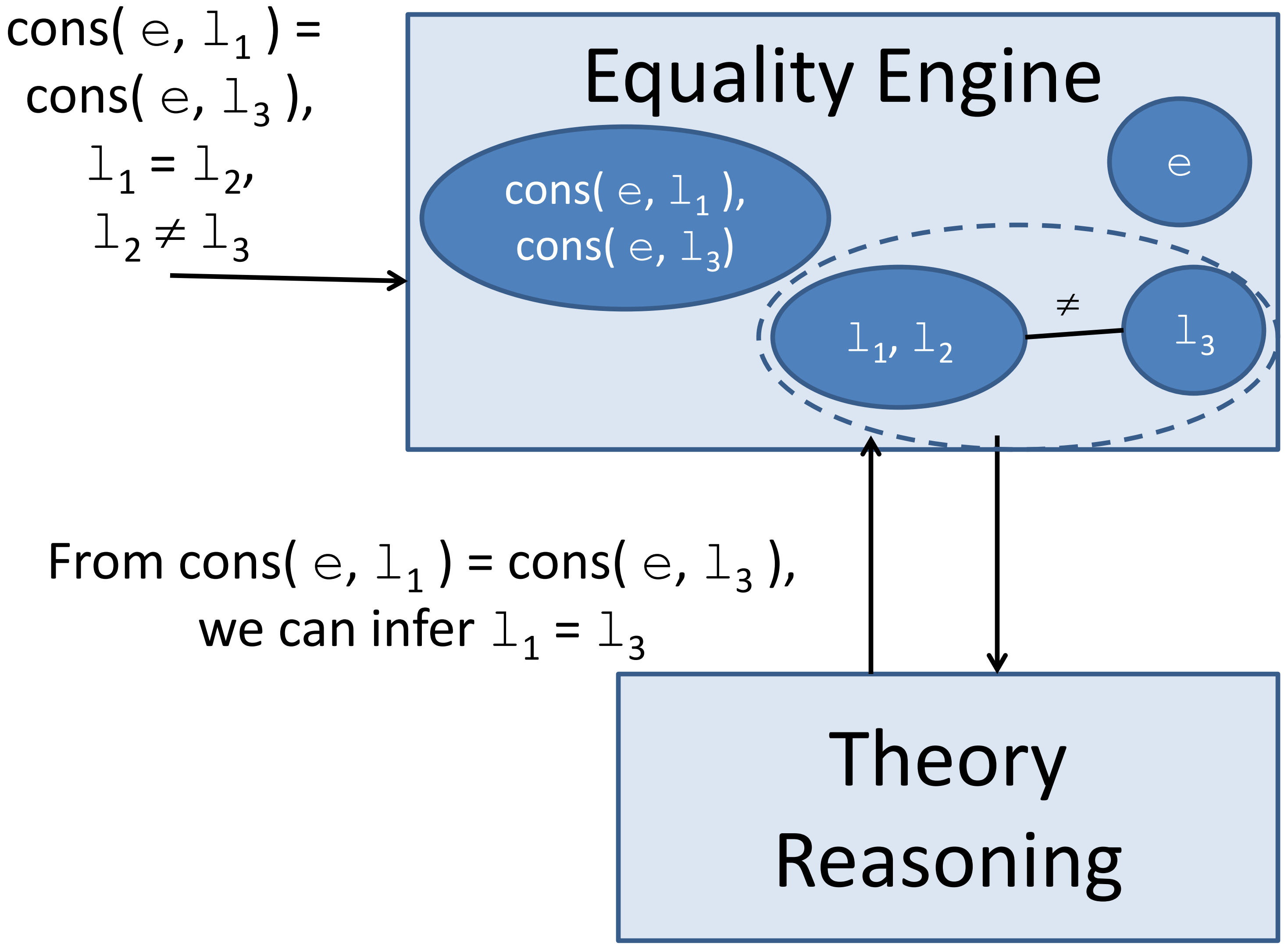


Case : Inductive Datatypes

cons(e, l_1) =
cons(e, l_3),
$l_1 = l_2$,
$l_2 \neq l_3$

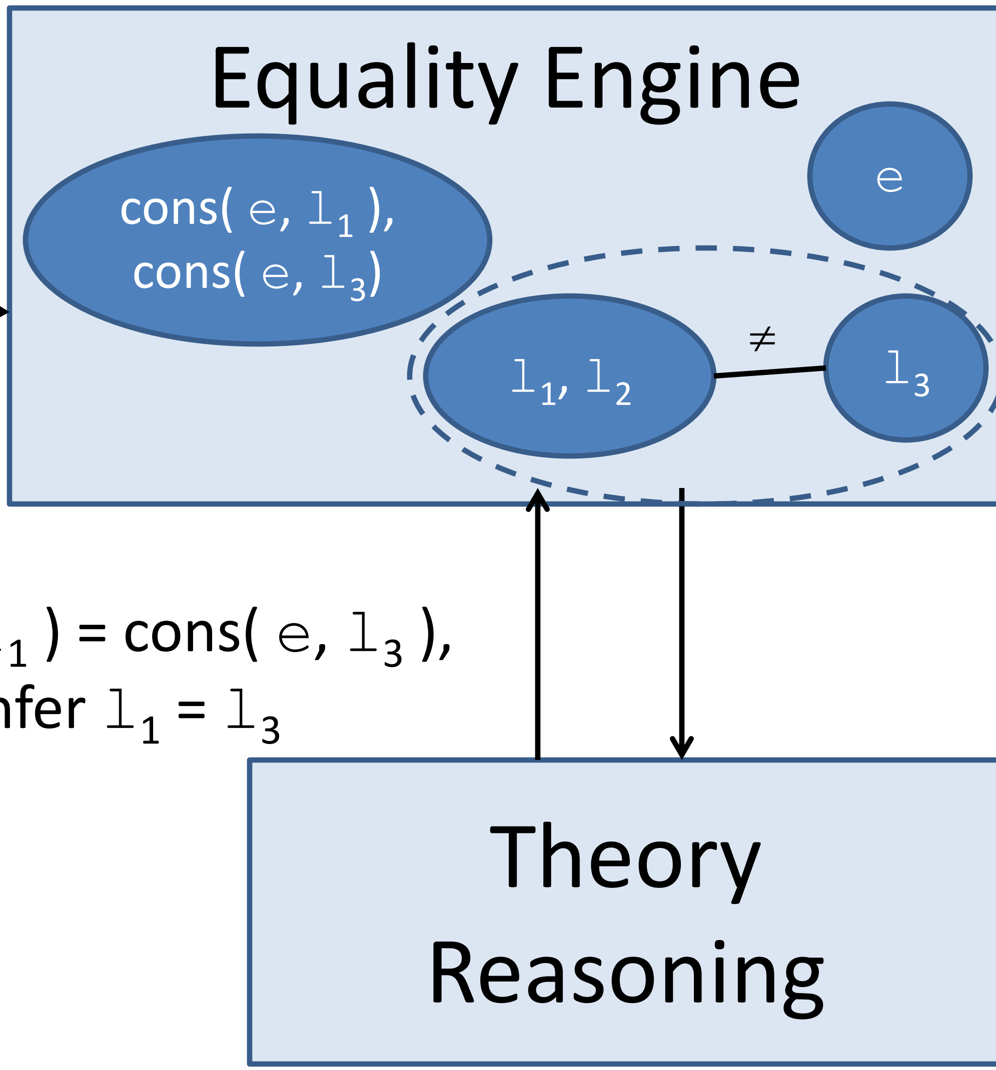


Case : Inductive Datatypes



Case : Inductive Datatypes

cons(e, l_1) = cons(e, l_3), $l_1 = l_2$, $l_2 \neq l_3$



From $l_1 = l_3$, $l_1 \neq l_3$

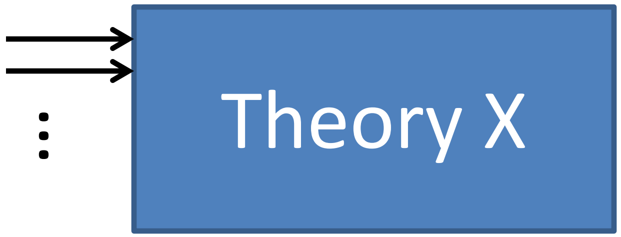
We have conflict : cons(e, l_1) = cons(e, l_3), $l_1 = l_2$, $l_2 \neq l_3$

From cons(e, l_1) = cons(e, l_3), we can infer $l_1 = l_3$

Theory Solver (summary)

- Most theory solvers rely on Equality Engine for:
  - Computing equivalence classes of current terms
  - Reporting most conflicts
  - Performing (eager) T-propagation
- Supplement with Theory Reasoning :
  - Adds assertions inferred from current state
  - May add other lemmas to system when necessary

Theory Interface (extended)



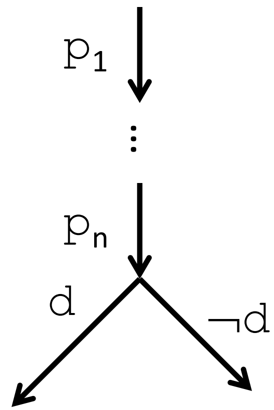
- In addition to `check` if assertions T-consistent,
  - `propagate`, T-propagate literals
  - `explain`, explain why literals were T-propagated
  - `collectModelInfo`, get model for curr assertions
  - Others:
    - `getNextDecision`
    - `staticLearn`
    - `preSolve`
    - …

Support for Theory Development

- Equality Engine data structure
- Associated "kinds" file
  - Contains specifications for:
    - Signature Definition (symbols in the theory)
    - Term normalization
    - Type checking
    - Properties of the theory
      - Interaction when performing theory combination
  - Auto-generates necessary code for each of these
- Automatic Integration into Theory Engine

Questions?

DPLL(T) Search : Incremental Checking



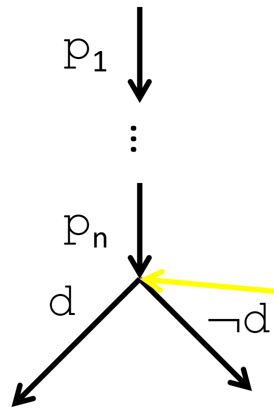
Propagate : Literals p_1,…,p_n must be added to M

Decide : Add literal d to M

DPLL(T) Search : Incremental Checking

Propagate : Literals $p_1, \ldots, p_n$ must be added to M

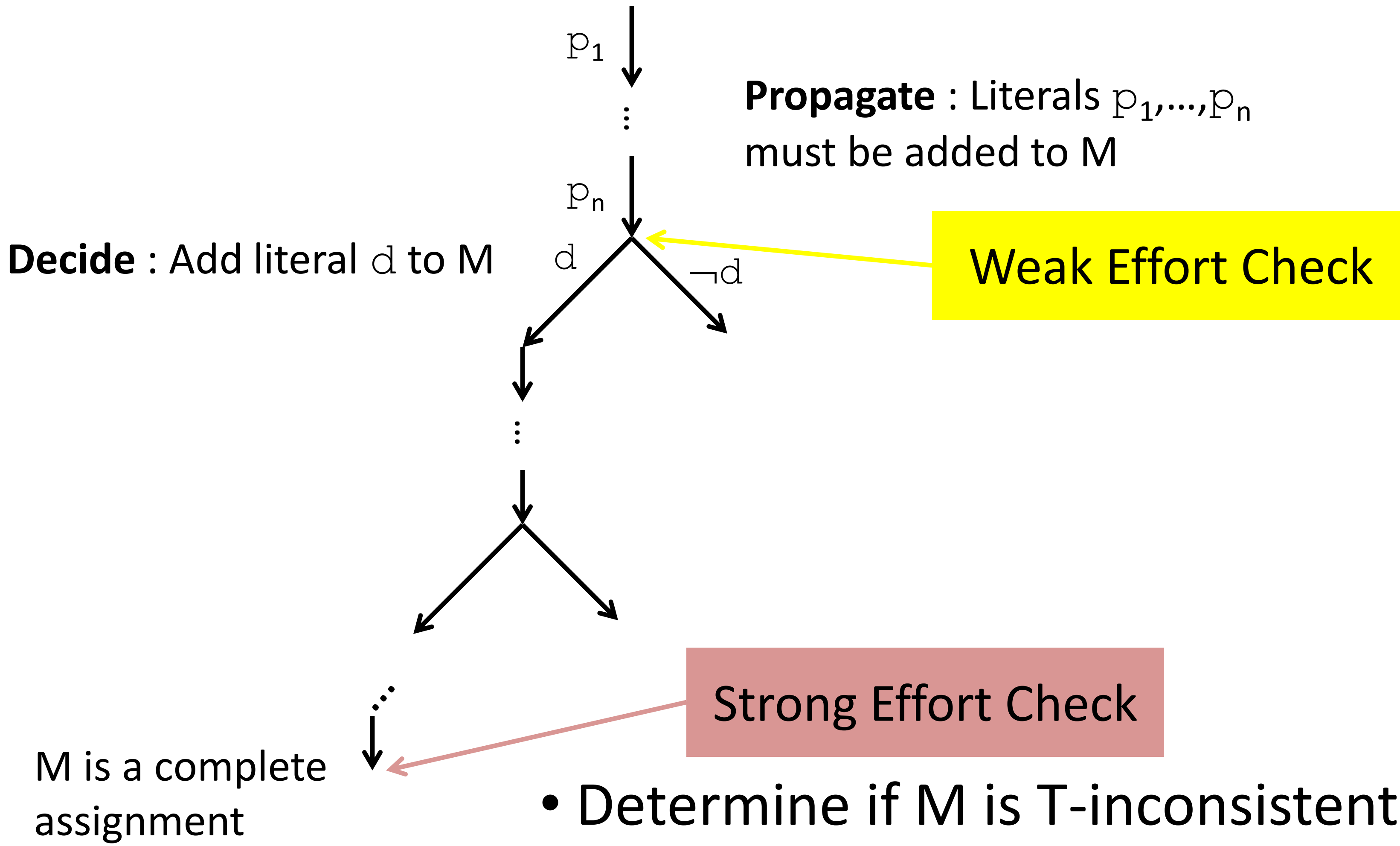
Decide : Add literal d to M



Weak Effort Check

- Check if $p_1 \ldots p_n$ are already T-inconsistent
- Should be efficient
- Can be incomplete

DPLL(T) Search : Incremental Checking



- Determine if M is T-inconsistent
- Must be complete