CS 292C Computer-Aided Reasoning for Software

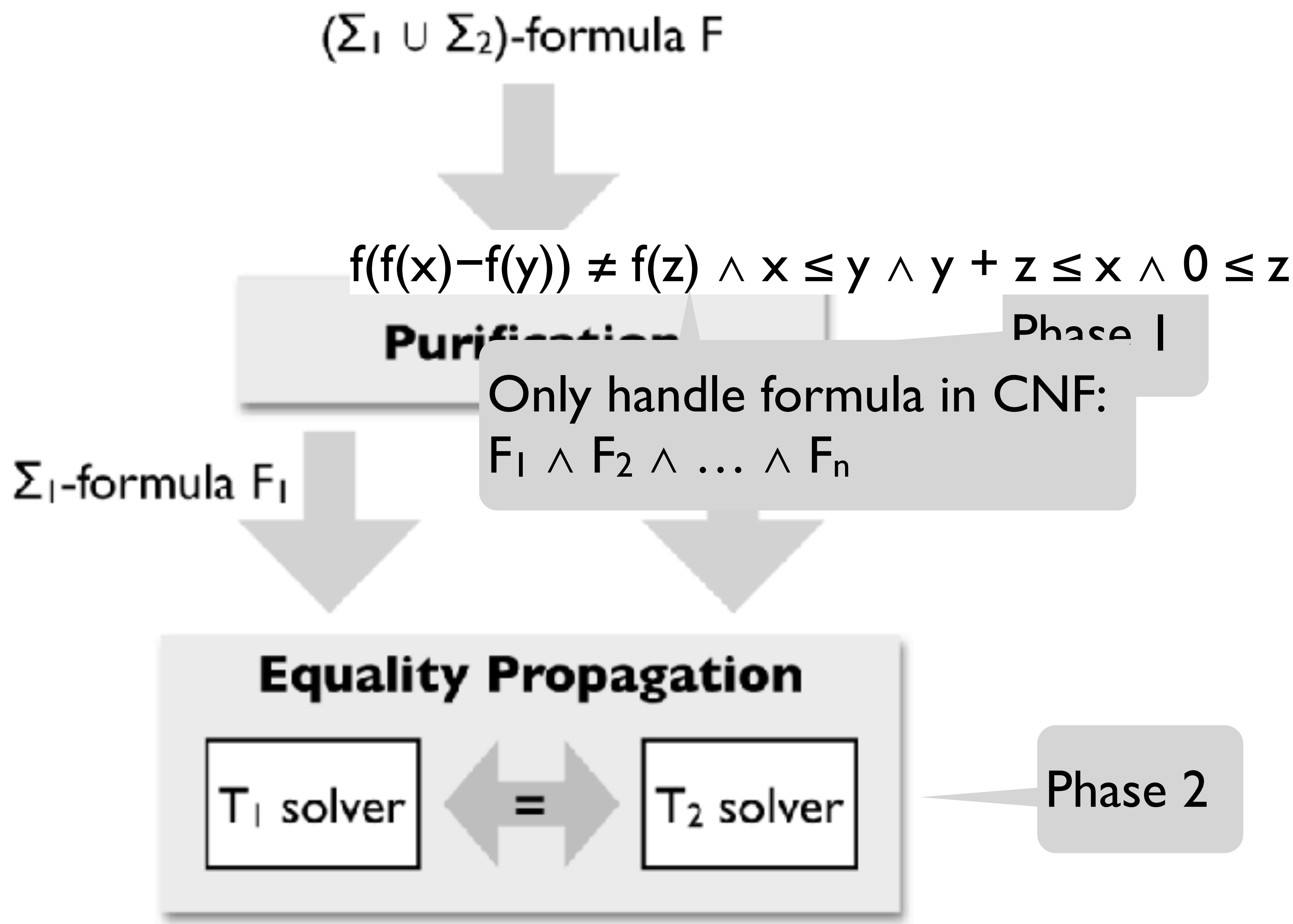
Lecture 9: The DPLL(T) Framework

Yu Feng
Spring 2024

Summary of previous lecture

- 4th paper review is out
- Proposal was graded now
- Deciding a combination of theories
- The Nelson-Oppen algorithm

Overview of Nelson-Oppen



Outline of this lecture

- Deciding arbitrary boolean combinations of theory constraints
- The DPLL (T) algorithm
- The last lecture about SMT/SAT

Boolean abstraction

CFG of SMT formula in theory T

- $F := a_T \mid F_1 \wedge F_2 \mid F_1 \vee F_2 \mid \neg F$

For each SMT formula, define a **boolean abstraction function**, that maps SMT formula to overapproximate SAT formula

- $B(a_T) = b$ (b fresh)
- $B(F_1 \wedge F_2) = B(F_1) \wedge B(F_2)$
- $B(F_1 \vee F_2) = B(F_1) \vee B(F_2)$
- $B(\neg F) = \neg B(F_1)$

$F : x = z \wedge ((y = z \wedge x < z) \vee \neg(x = z))$

$B(F) = b_1 \wedge ((b_2 \wedge b_3) \vee \neg b_1)$

Is B(F) satisfiable?
Is F satisfiable?

Off-line v.s. online

SAT solver may yield assignments that are not sat modulo T because boolean abstraction is an over-approximation

Need to learn theory conflict clauses

Two different approaches for learning theory conflict clauses

- Off-line (eager): Use SAT solver as black-box
- On-line (lazy): Integrate theory solver into the CDCL loop (adopted by mainstream SMT solvers)

Off-line version

Offline-DPLL(T-formula φ)

$\varphi_P \leftarrow$ B(φ)

while (TRUE) **do**

  μ_P, res $\leftarrow$ **CDCL**(φ_P)

  if res = UNSAT **then return** UNSAT

  else

    T-res $\leftarrow$ T-solve(**B^{-1}**(μ_P))

    if T-res = SAT **then return** SAT

    else $\varphi_P \leftarrow \varphi_P \wedge \neg \mu_P$

F : $x = z \wedge ((y = z \wedge x < z) \vee \neg(x = z))$

$B(F) = b_1 \wedge ((b_2 \wedge b_3) \vee \neg b_1)$

SAT assignment to B(F): $b_1 \wedge b_2 \wedge b_3$

$B^{-1}(F)$ is UNSAT

What is new boolean abstraction?

$b_1 \wedge ((b_2 \wedge b_3) \vee \neg b_1) \wedge \neg(b_1 \wedge b_2 \wedge b_3)$

Is this formula SAT?

$B^{-1}(F) = x = y \wedge x < y \wedge a_1 \wedge a_2 \wedge \ldots \wedge a_{2019}$

2^{2019} UNSAT assignments containing

$x = y \wedge x < y$ but $\neg A$ prevents only one of them

Theory conflict clause

Minimal UNSAT core

- Let ϕ be original unsatisfiable conjunct
- Drop one atom from ϕ, call this ϕ'
- If ϕ' is still unsat, $\phi := \phi'$
- Repeat this for every atom in ϕ
- resulting ϕ is minimal unsat core of original formula

$\phi : x = y \wedge f(x)+z = 5 \wedge f(x) \neq f(y) \wedge y \leq 3$

Drop $x = y$ from ϕ. Is result UNSAT?

Drop $f(x)+z = 5$. Is result UNSAT?

New formula: $\phi : x = y \wedge f(x) \neq f(y) \wedge y \leq 3$

Drop $f(x) \neq f(y)$. Is result UNSAT?

Drop $y \leq 3$. Is result UNSAT?

So, minimal UNSAT core is $x = y \wedge f(x) \neq f(y)$

Improvement on the off-line

Offline-DPLL(T-formula φ)

$\varphi_P \leftarrow$ B(φ)

while (TRUE) **do**

  μ_P, res $\leftarrow$ **CDCL**(φ_P)

  if res = UNSAT **then return** UNSAT

  else

    T-res $\leftarrow$ T-solve(**B⁻¹**(μ_P))

    if T-res = SAT **then return** SAT

    else

      t $\leftarrow$ **B**(UNSATCORE(**B⁻¹**(μ_P)))

      $\varphi_P \leftarrow \varphi_P \wedge \neg t$

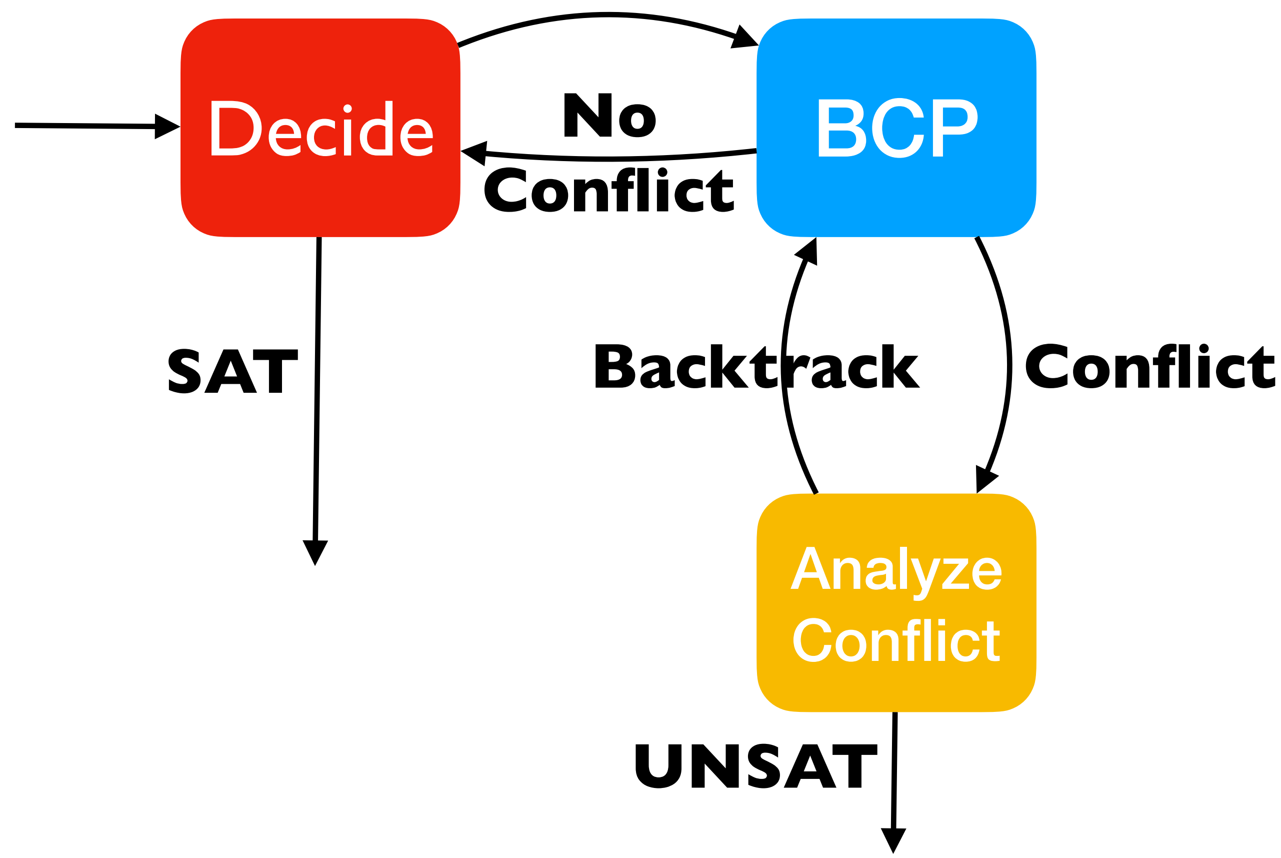
$B^{-1}(F) = x = y \wedge x < y \wedge a_1 \wedge a_2 \wedge ... \wedge a_{2019}$

x = y and x < y are overapproximated by boolean variables b_1 and b_2

we are doomed if both b_1 and b_2 are true.

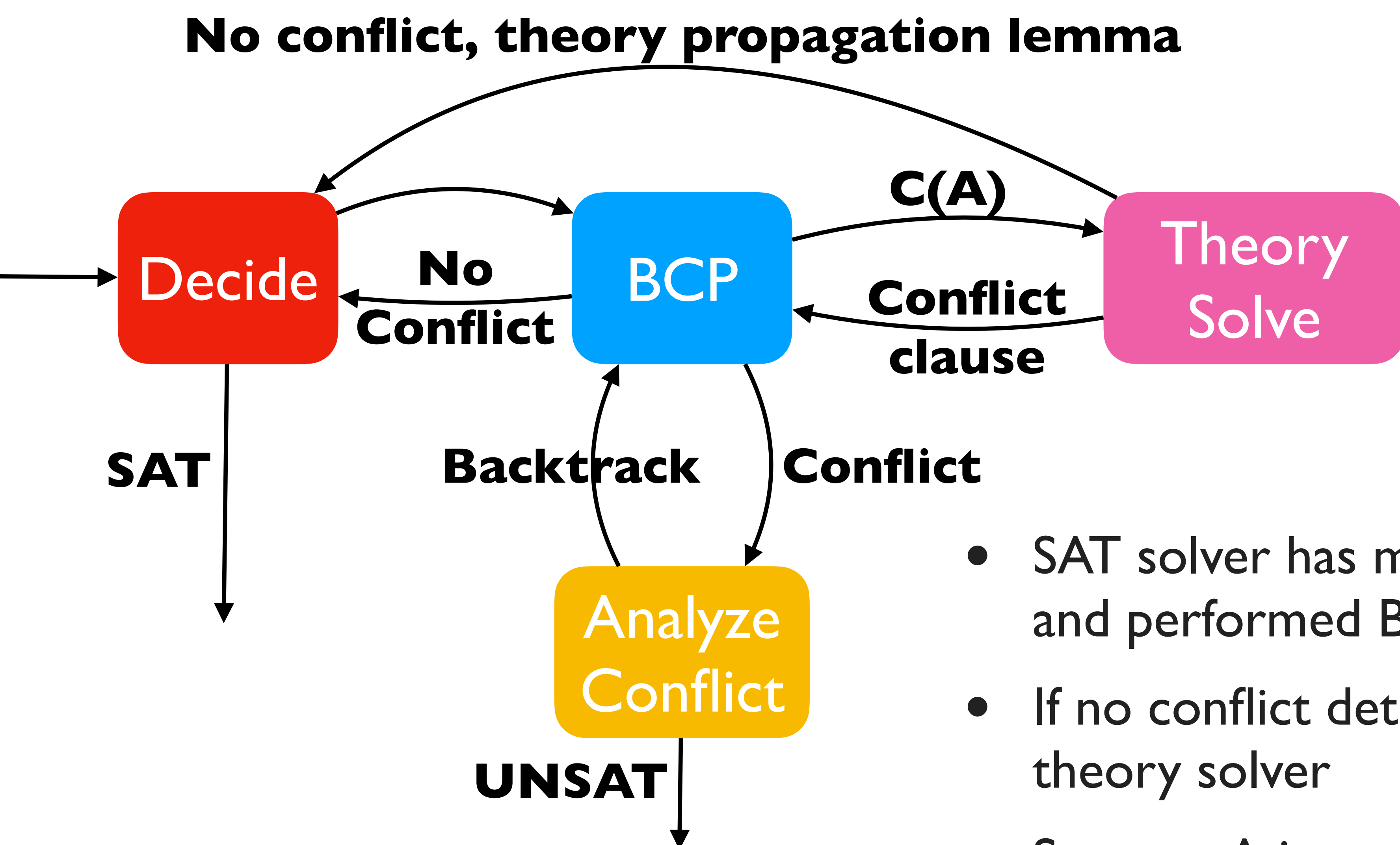
Better but still need a *full assignment* to the boolean abstraction in order to generate a conflict clause.

DPLL-based SAT solver



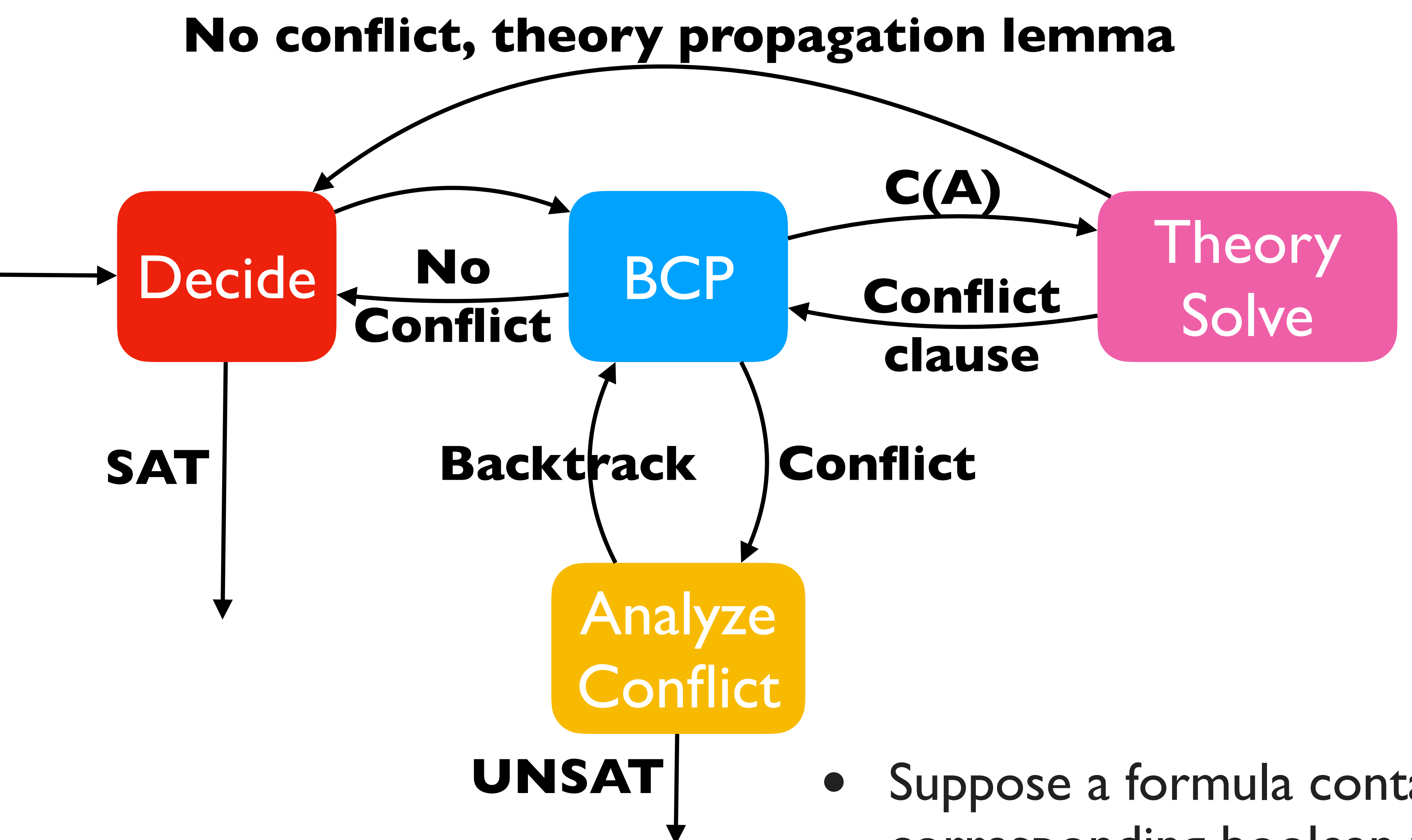
Integrate theory solver right into this SAT solving loop!

From DPLL to DPLL(T)



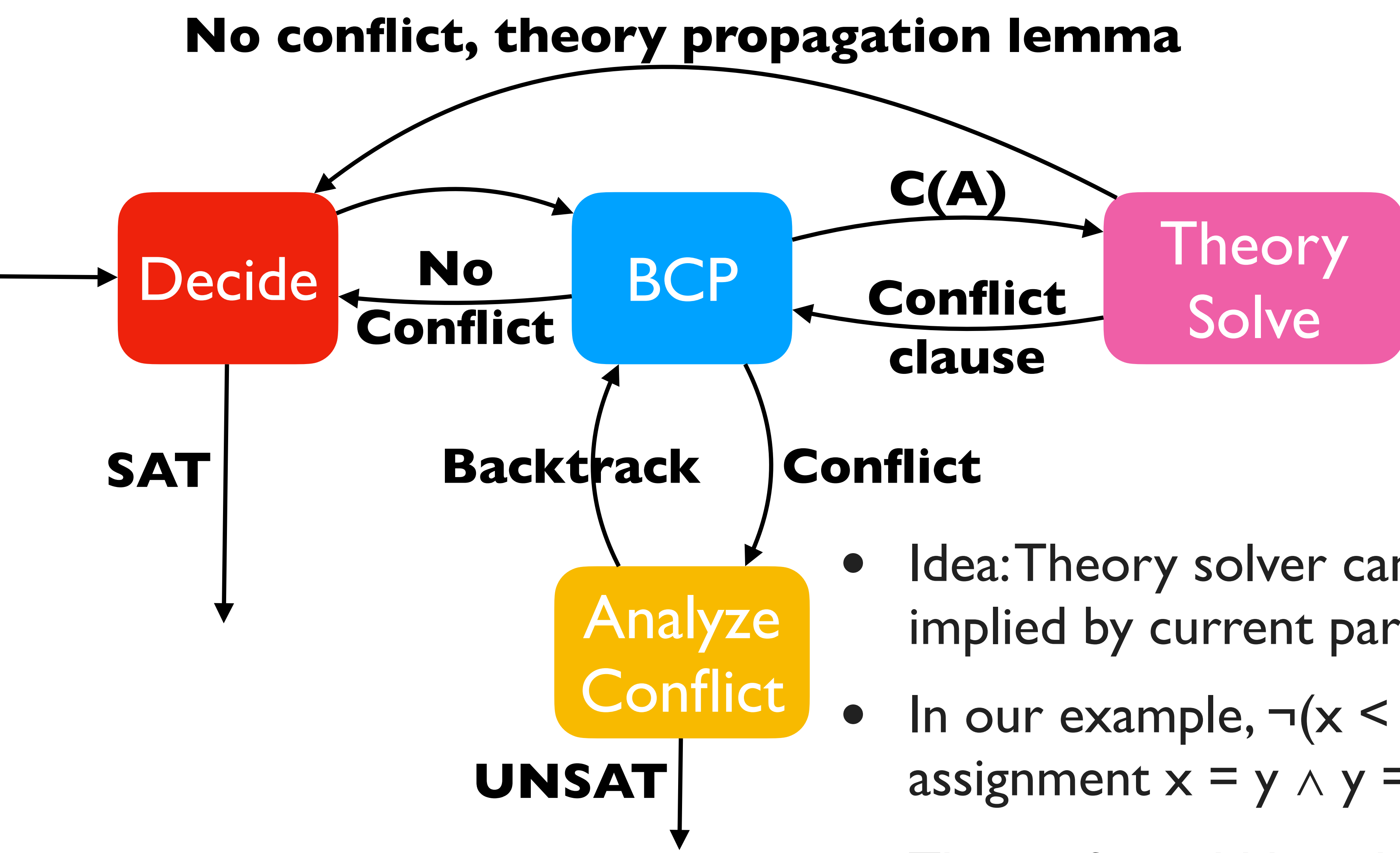
- SAT solver has made assignment in Decide step and performed BCP
- If no conflict detected, immediately invoke theory solver
- Suppose A is current partial assignment to boolean abstraction
- Use theory solver to decide if $B^{-1}(A)$ is UNSAT
- If $B^{-1}(A)$ UNSAT, add theory conflict clause ¬A to clause database

Theory Propagation Lemmas



- Suppose a formula contains $x = y, y = z, x < z$ with corresponding boolean variables b_1, b_2, b_3
- Suppose SAT solver makes partial assignment $b_1: \top, b_2: \top$
- In next Decide step, free to assign $b_3: \top$ or $b_3: \bot$
- But assignment $b_3: \top$ is stupid, why?

Theory Propagation Lemmas



- Idea: Theory solver can communicate which literals are implied by current partial assignment
- In our example, ¬(x < z) implied by current partial assignment $x = y \land y = z$
- Thus, safely add $b_1 \land b_2 \rightarrow \neg b_3$ to clause database
- The clauses implied by theory are theory propagation lemmas

The lemmas prevents bad assignments to boolean abstraction

Theory Propagation Lemmas

- Which theory propagation lemmas do we add?
  - Option #1 (exhaustive): Figure out and add all literals implied by current partial assignment
  - Option #2 (heuristics): Only figure out literals "obviously" implied by current partial assignment
- Exhaustive theory propagation can be very expensive
- There isn't much of a science behind which literals are "obviously" implied
- Solvers use different heuristics to obtain simple-to-find implications

Modern SMT solvers

- All competitive SMT solvers today are based on the on-line version
- Many existing off-the-shelf SMT solvers: Z3, CVC4, Yices, MathSAT, etc.
- Lots of on-going research on SMT, esp. related to quantifier support
- Annual competition SMT-COM

TODOs by next lecture

- Start to work on your final report