Computer-Aided Reasoning for Software

CSE507

The DPLL(T) Framework

Emina Torlak
emina@cs.washington.edu

Today

Last lecture

- Deciding conjunctions of $(T_1 \cup T_2)$-constraints with Nelson-Oppen

Today

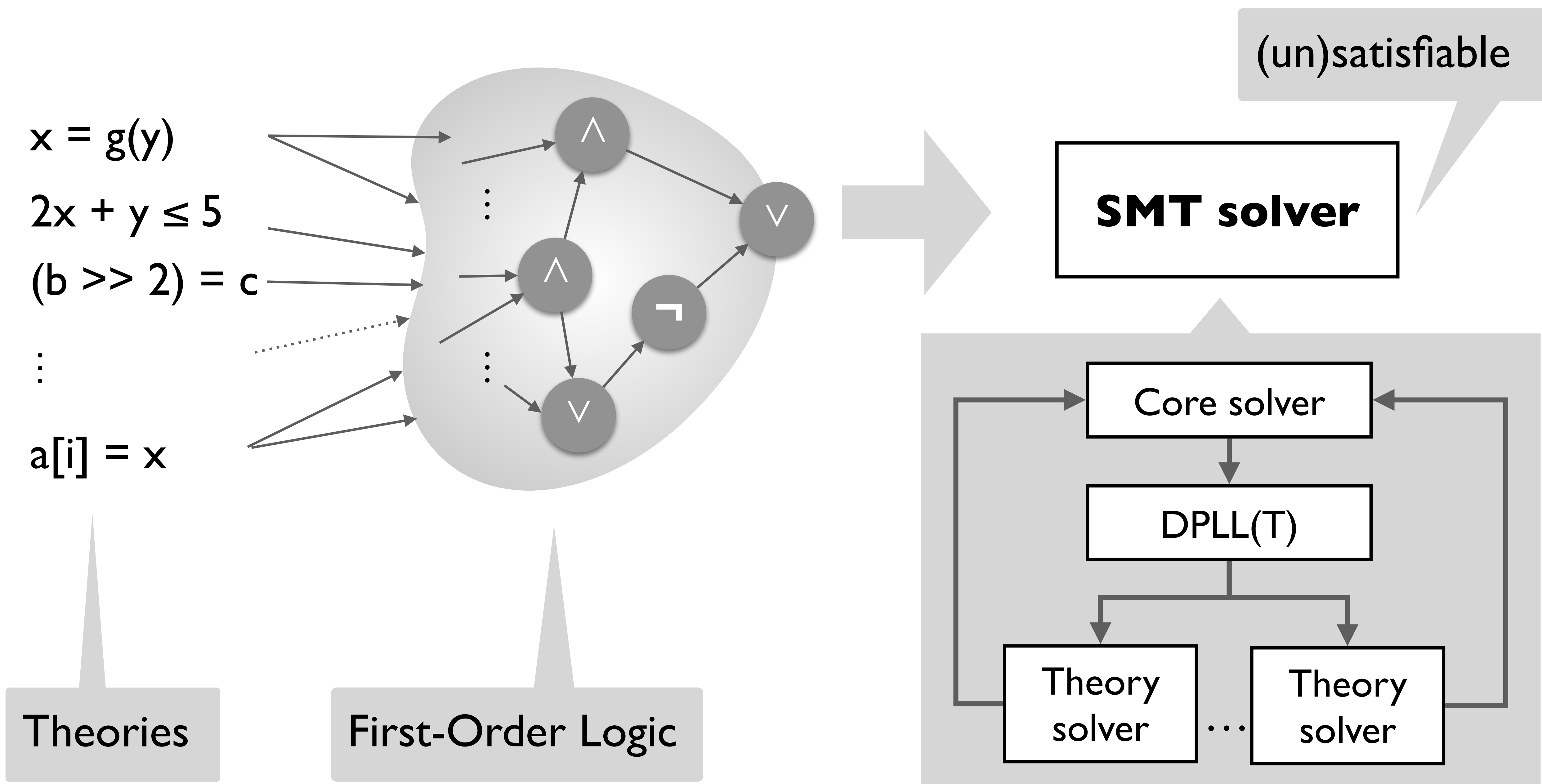
- Deciding arbitrary boolean combinations of theory constraints

Reminders

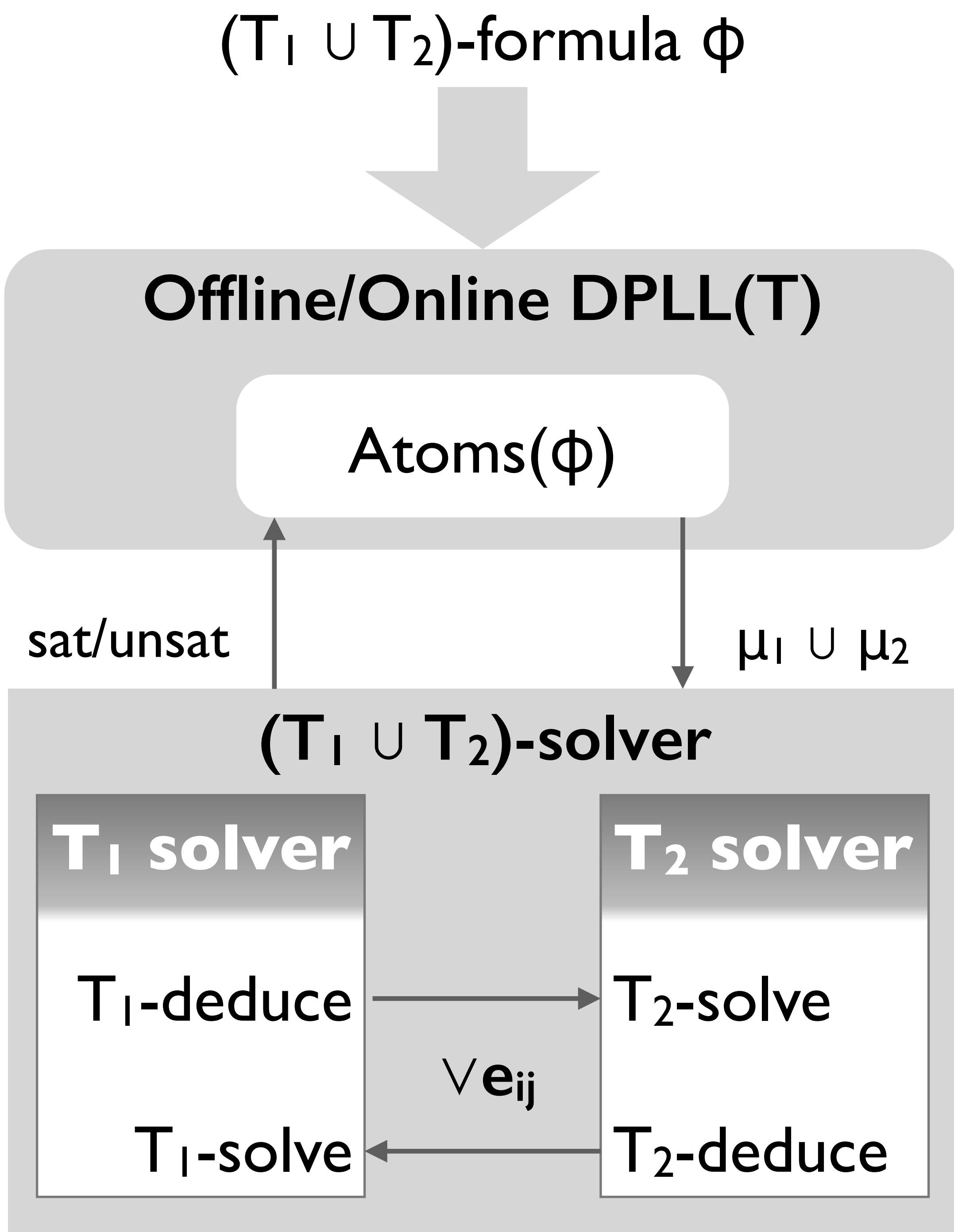
- Project proposal due tonight.

topics

Recall: Satisfiability Modulo Theories (SMT)

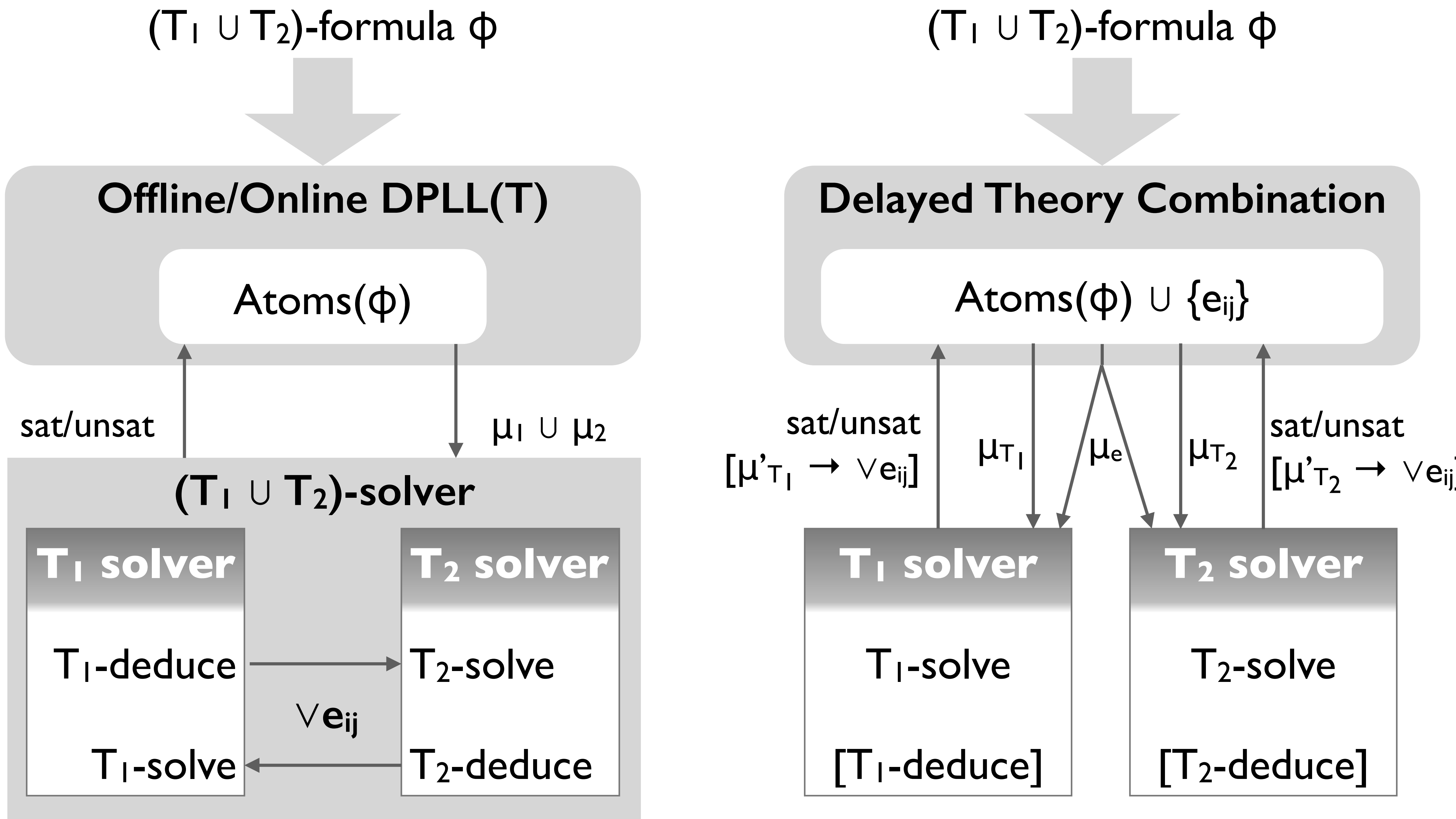


The DPLL(T) Framework



R. Bruttomesso et al., DTC vs NO, 2009

The DPLL(T) Framework



R. Bruttomesso et al., DTC vs NO, 2009

Offline DPLL(T)

```
Offline-DPLL_T(T-formula φ)
  φᴾ ← T2B(φ)
  while (TRUE) do
    μᴾ, res ← CDCL(φᴾ)
    if res = UNSAT then return UNSAT
    else
      T-res ← T-solve(B2T(μᴾ))
      if T-res = SAT then return SAT
      else φᴾ ← φᴾ ∧ ¬μᴾ
```

Offline DPLL(T)

Assume ϕ is in CNF.

Offline-DPLL$_T$(T-formula ϕ)
  $\phi^P \leftarrow$ **T2B**(ϕ)
  while (TRUE) **do**
    μ^P, res $\leftarrow$ **CDCL**(ϕ^P)
    if res = UNSAT **then return** UNSAT
    else
      T-res $\leftarrow$ T-solve(**B2T**(μ^P))
      if T-res = SAT **then return** SAT
      else $\phi^P \leftarrow \phi^P \wedge \neg\mu^P$

Offline DPLL(T)

```
Offline-DPLL_T(T-formula φ)
  φP ← T2B(φ)
  while (TRUE) do
    μP, res ← CDCL(φP)
    if res = UNSAT then return UNSAT
    else
      T-res ← T-solve(B2T(μP))
      if T-res = SAT then return SAT
      else φP ← φP ∧ ¬μP
```

Assume φ is in CNF.

T2B computes the *boolean abstraction* (aka *boolean skeleton*) of φ by replacing every atom in φ with a fresh boolean variable.

Offline DPLL(T)

Offline-DPLL$_T$(T-formula ϕ)
  $\phi^P \leftarrow$ **T2B**(ϕ)
  while (TRUE) **do**
    μ^P, res $\leftarrow$ **CDCL**(ϕ^P)
    if res = UNSAT **then return** UNSAT
    else
      T-res $\leftarrow$ T-solve(**B2T**(μ^P))
      if T-res = SAT **then return** SAT
      else $\phi^P \leftarrow \phi^P \wedge \neg\mu^P$

Assume ϕ is in CNF.

T2B computes the *boolean abstraction* (aka *boolean skeleton*) of ϕ by replacing every atom in ϕ with a fresh boolean variable.

B2T computes the *boolean refinement* of the current propositional assignment μ^P.

Offline DPLL(T)

```
Offline-DPLLT(T-formula ϕ)
  ϕP ← T2B(ϕ)
  while (TRUE) do
    μP, res ← CDCL(ϕP)
    if res = UNSAT then return UNSAT
    else
      T-res ← T-solve(B2T(μP))
      if T-res = SAT then return SAT
      else ϕP ← ϕP ∧ ¬μP
```

Assume ϕ is in CNF.

T2B computes the *boolean abstraction* (aka *boolean skeleton*) of ϕ by replacing every atom in ϕ with a fresh boolean variable.

B2T computes the *boolean refinement* of the current propositional assignment μ^P.

Theory conflict clause.

Boolean abstraction (T2B) and refinement (B2T)

T2B(ϕ)

- T2B(a_i) = b_i, if a_i is a theory atom and b_i is a fresh boolean variable
- T2B(b_j) = b_j, if b_j is a boolean variable
- T2B($\phi_1 \wedge \phi_2$) = T2B(ϕ_1) $\wedge$ T2B(ϕ_2)
- T2B($\phi_1 \vee \phi_2$) = T2B(ϕ_1) $\vee$ T2B(ϕ_2)
- T2B($\neg\phi_1$) = $\neg$T2B(ϕ_1)

B2T(ϕ^p) = T2B^{-1}(ϕ^p)

Boolean abstraction (T2B) and refinement (B2T)

T2B(ϕ)

- T2B(a_i) = b_i, if a_i is a theory atom and b_i is a fresh boolean variable
- T2B(b_j) = b_j, if b_j is a boolean variable
- T2B($\phi_1 \wedge \phi_2$) = T2B(ϕ_1) $\wedge$ T2B(ϕ_2)
- T2B($\phi_1 \vee \phi_2$) = T2B(ϕ_1) $\vee$ T2B(ϕ_2)
- T2B($\neg\phi_1$) = $\neg$T2B(ϕ_1)

ϕ: $(x = 1) \wedge ((x = 2) \vee (x = 3))$

B2T(ϕ^p) = T2B^{-1}(ϕ^p)

Boolean abstraction (T2B) and refinement (B2T)

T2B(ϕ)

- $T2B(a_i) = b_i$, if a_i is a theory atom and b_i is a fresh boolean variable
- $T2B(b_j) = b_j$, if b_j is a boolean variable
- $T2B(\phi_1 \wedge \phi_2) = T2B(\phi_1) \wedge T2B(\phi_2)$
- $T2B(\phi_1 \vee \phi_2) = T2B(\phi_1) \vee T2B(\phi_2)$
- $T2B(\neg\phi_1) = \neg T2B(\phi_1)$

ϕ: $(x = 1) \wedge ((x = 2) \vee (x = 3))$

$T2B(\phi)$: $b_1 \wedge (b_2 \vee b_3)$

$B2T(\phi^p) = T2B^{-1}(\phi^p)$

Boolean abstraction (T2B) and refinement (B2T)

T2B(ϕ)

- T2B(a_i) = b_i, if a_i is a theory atom and b_i is a fresh boolean variable
- T2B(b_j) = b_j, if b_j is a boolean variable
- T2B($\phi_1 \wedge \phi_2$) = T2B(ϕ_1) $\wedge$ T2B(ϕ_2)
- T2B($\phi_1 \vee \phi_2$) = T2B(ϕ_1) $\vee$ T2B(ϕ_2)
- T2B($\neg\phi_1$) = $\neg$T2B(ϕ_1)

ϕ: $(x = 1) \wedge ((x = 2) \vee (x = 3))$

T2B(ϕ): $b_1 \wedge (b_2 \vee b_3)$

B2T($b_1 \wedge b_3$): $(x = 1) \wedge (x = 3)$

B2T(ϕ^p) = T2B^{-1}(ϕ^p)

Soundness and termination of offline DPLL(T)

Offline-DPLL$_T$(T-formula ϕ)
  $\phi^P \leftarrow$ **T2B**(ϕ)
  while (TRUE) **do**
    μ^P, res $\leftarrow$ **CDCL**(ϕ^P)
    if res = UNSAT **then return** UNSAT
    else
      T-res $\leftarrow$ T-solve(**B2T**(μ^P))
      if T-res = SAT **then return** SAT
      else $\phi^P \leftarrow \phi^P \wedge \neg\mu^P$

Soundness and termination of offline DPLL(T)

Offline-DPLL$_T$(T-formula ϕ)
  $\phi^P \leftarrow$ **T2B**(ϕ)
  while (TRUE) **do**
    μ^P, res $\leftarrow$ **CDCL**(ϕ^P)
    if res = UNSAT **then return** UNSAT
    else
      T-res $\leftarrow$ T-solve(**B2T**(μ^P))
      if T-res = SAT **then return** SAT
      else $\phi^P \leftarrow \phi^P \land \neg\mu^P$

The algorithm is sound because ϕ^P overapproximates the satisfiability of ϕ at the beginning of each loop iteration (so it never returns UNSAT for SAT formulas), and it only returns SAT when μ^P both satisfies ϕ^P and is consistent with the theory axioms.

Soundness and termination of offline DPLL(T)

```
Offline-DPLL_T(T-formula φ)
  φP ← T2B(φ)
  while (TRUE) do
    μP, res ← CDCL(φP)
    if res = UNSAT then return UNSAT
    else
      T-res ← T-solve(B2T(μP))
      if T-res = SAT then return SAT
      else φP ← φP ∧ ¬μP
```

The algorithm is sound because ϕ^P overapproximates the satisfiability of ϕ at the beginning of each loop iteration (so it never returns UNSAT for SAT formulas), and it only returns SAT when μ^P both satisfies ϕ^P and is consistent with the theory axioms.

The algorithm terminates because there are finitely many satisfying assignments to the boolean abstraction, and we get a different one every time.

Offline DPLL(T) example

Offline-DPLL$_T$(T-formula ϕ)
  $\phi^P \leftarrow$ **T2B**(ϕ)
  while (TRUE) **do**
    μ^P, res $\leftarrow$ **CDCL**(ϕ^P)
    if res = UNSAT **then return** UNSAT
    else
      T-res $\leftarrow$ T-solve(**B2T**(μ^P))
      if T-res = SAT **then return** SAT
      else $\phi^P \leftarrow \phi^P \wedge \neg\mu^P$

Offline DPLL(T) example

```
Offline-DPLL_T(T-formula φ)
  φP ← T2B(φ)
  while (TRUE) do
    μP, res ← CDCL(φP)
    if res = UNSAT then return UNSAT
    else
      T-res ← T-solve(B2T(μP))
      if T-res = SAT then return SAT
      else φP ← φP ∧ ¬μP
```

- $\phi \leftarrow (x = 1) \wedge ((x = 2) \vee (x = 3))$

Offline DPLL(T) example

Offline-DPLL$_T$(T-formula ϕ)
  $\phi^P \leftarrow$ **T2B**(ϕ)
  while (TRUE) **do**
    μ^P, res $\leftarrow$ **CDCL**(ϕ^P)
    if res = UNSAT **then return** UNSAT
    else
      T-res $\leftarrow$ T-solve(**B2T**(μ^P))
      if T-res = SAT **then return** SAT
      else $\phi^P \leftarrow \phi^P \wedge \neg\mu^P$

- $\phi \leftarrow (x = 1) \wedge ((x = 2) \vee (x = 3))$
- $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3)$

Offline DPLL(T) example

Offline-DPLL$_T$(T-formula ϕ)
  $\phi^P \leftarrow$ **T2B**(ϕ)
  while (TRUE) **do**
    μ^P, res $\leftarrow$ **CDCL**(ϕ^P)
    if res = UNSAT **then return** UNSAT
    else
      T-res $\leftarrow$ T-solve(**B2T**(μ^P))
      if T-res = SAT **then return** SAT
      else $\phi^P \leftarrow \phi^P \wedge \neg\mu^P$

- $\phi \leftarrow (x = 1) \wedge ((x = 2) \vee (x = 3))$
- $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3)$
- $\mu^P \leftarrow b_1 \wedge b_2 \wedge b_3$

Offline DPLL(T) example

Offline-DPLL$_T$(T-formula ϕ)
  $\phi^P \leftarrow$ **T2B**(ϕ)
  while (TRUE) **do**
    μ^P, res $\leftarrow$ **CDCL**(ϕ^P)
    if res = UNSAT **then return** UNSAT
    else
      T-res $\leftarrow$ T-solve(**B2T**(μ^P))
      if T-res = SAT **then return** SAT
      else $\phi^P \leftarrow \phi^P \wedge \neg\mu^P$

- $\phi \leftarrow (x = 1) \wedge ((x = 2) \vee (x = 3))$
- $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3)$
- $\mu^P \leftarrow b_1 \wedge b_2 \wedge b_3$
  - $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_2 \vee \neg b_3)$

Offline DPLL(T) example

Offline-DPLL$_T$(T-formula ϕ)
  $\phi^P \leftarrow$ **T2B**(ϕ)
  while (TRUE) **do**
    μ^P, res $\leftarrow$ **CDCL**(ϕ^P)
    if res = UNSAT **then return** UNSAT
    else
      T-res $\leftarrow$ T-solve(**B2T**(μ^P))
      if T-res = SAT **then return** SAT
      else $\phi^P \leftarrow \phi^P \wedge \neg\mu^P$

- $\phi \leftarrow (x = 1) \wedge ((x = 2) \vee (x = 3))$
- $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3)$
- $\mu^P \leftarrow b_1 \wedge b_2 \wedge b_3$
    - $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_2 \vee \neg b_3)$
- $\mu^P \leftarrow b_1 \wedge \neg b_2 \wedge b_3$

Offline DPLL(T) example

Offline-DPLL$_T$(T-formula ϕ)
  $\phi^P \leftarrow$ **T2B**(ϕ)
  while (TRUE) **do**
    μ^P, res $\leftarrow$ **CDCL**(ϕ^P)
    if res = UNSAT **then return** UNSAT
    else
      T-res $\leftarrow$ T-solve(**B2T**(μ^P))
      if T-res = SAT **then return** SAT
      else $\phi^P \leftarrow \phi^P \wedge \neg\mu^P$

- $\phi \leftarrow (x = 1) \wedge ((x = 2) \vee (x = 3))$
- $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3)$
- $\mu^P \leftarrow b_1 \wedge b_2 \wedge b_3$
  - $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_2 \vee \neg b_3)$
- $\mu^P \leftarrow b_1 \wedge \neg b_2 \wedge b_3$
  - $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_2 \vee \neg b_3) \wedge (\neg b_1 \vee b_2 \vee \neg b_3)$

Offline DPLL(T) example

Offline-DPLL$_T$(T-formula ϕ)
  $\phi^P \leftarrow$ **T2B**(ϕ)
  while (TRUE) **do**
    μ^P, res $\leftarrow$ **CDCL**(ϕ^P)
    if res = UNSAT **then return** UNSAT
    else
      T-res $\leftarrow$ T-solve(**B2T**(μ^P))
      if T-res = SAT **then return** SAT
      else $\phi^P \leftarrow \phi^P \wedge \neg\mu^P$

- $\phi \leftarrow (x = 1) \wedge ((x = 2) \vee (x = 3))$
- $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3)$
- $\mu^P \leftarrow b_1 \wedge b_2 \wedge b_3$
  - $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_2 \vee \neg b_3)$
- $\mu^P \leftarrow b_1 \wedge \neg b_2 \wedge b_3$
  - $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_2 \vee \neg b_3) \wedge (\neg b_1 \vee b_2 \vee \neg b_3)$
- $\mu^P \leftarrow b_1 \wedge b_2 \wedge \neg b_3$

Offline DPLL(T) example

```
Offline-DPLL_T(T-formula ϕ)
  ϕP ← T2B(ϕ)
  while (TRUE) do
    μP, res ← CDCL(ϕP)
    if res = UNSAT then return UNSAT
    else
      T-res ← T-solve(B2T(μP))
      if T-res = SAT then return SAT
      else ϕP ← ϕP ∧ ¬μP
```

- $\phi \leftarrow (x = 1) \wedge ((x = 2) \vee (x = 3))$
- $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3)$
- $\mu^P \leftarrow b_1 \wedge b_2 \wedge b_3$
  - $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_2 \vee \neg b_3)$
- $\mu^P \leftarrow b_1 \wedge \neg b_2 \wedge b_3$
  - $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_2 \vee \neg b_3) \wedge (\neg b_1 \vee b_2 \vee \neg b_3)$
- $\mu^P \leftarrow b_1 \wedge b_2 \wedge \neg b_3$
  - $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_2 \vee \neg b_3) \wedge (\neg b_1 \vee b_2 \vee \neg b_3) \wedge (\neg b_1 \vee \neg b_2 \vee b_3)$

Offline DPLL(T) example

```
Offline-DPLL_T(T-formula φ)
  φP ← T2B(φ)
  while (TRUE) do
    μP, res ← CDCL(φP)
    if res = UNSAT then return UNSAT
    else
      T-res ← T-solve(B2T(μP))
      if T-res = SAT then return SAT
      else φP ← φP ∧ ¬μP
```

- $\phi \leftarrow (x = 1) \wedge ((x = 2) \vee (x = 3))$
- $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3)$
- $\mu^P \leftarrow b_1 \wedge b_2 \wedge b_3$
  - $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_2 \vee \neg b_3)$
- $\mu^P \leftarrow b_1 \wedge \neg b_2 \wedge b_3$
  - $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_2 \vee \neg b_3) \wedge (\neg b_1 \vee b_2 \vee \neg b_3)$
- $\mu^P \leftarrow b_1 \wedge b_2 \wedge \neg b_3$
  - $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_2 \vee \neg b_3) \wedge (\neg b_1 \vee b_2 \vee \neg b_3) \wedge (\neg b_1 \vee \neg b_2 \vee b_3)$
- UNSAT

Offline DPLL(T): improvements

```
Offline-DPLL_T(T-formula ϕ)
  ϕP ← T2B(ϕ)
  while (TRUE) do
    μP, res ← CDCL(ϕP)
    if res = UNSAT then return UNSAT
    else
      T-res ← T-solve(B2T(μP))
      if T-res = SAT then return SAT
      else ϕP ← ϕP ∧ ¬μP
```

- $\phi \leftarrow (x = 1) \wedge ((x = 2) \vee (x = 3))$
- $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3)$
- $\mu^P \leftarrow b_1 \wedge b_2 \wedge b_3$
  - $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_2 \vee \neg b_3)$
- $\mu^P \leftarrow b_1 \wedge \neg b_2 \wedge b_3$
  - $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_2 \vee \neg b_3) \wedge (\neg b_1 \vee b_2 \vee \neg b_3)$
- $\mu^P \leftarrow b_1 \wedge b_2 \wedge \neg b_3$
  - $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_2 \vee \neg b_3) \wedge (\neg b_1 \vee b_2 \vee \neg b_3) \wedge (\neg b_1 \vee \neg b_2 \vee b_3)$
- UNSAT

Offline DPLL(T): improvements

Offline-DPLL$_T$(T-formula ϕ)
  $\phi^P \leftarrow$ **T2B**(ϕ)
  while (TRUE) **do**
    μ^P, res $\leftarrow$ **CDCL**(ϕ^P)
    if res = UNSAT **then return** UNSAT
    else
      T-res $\leftarrow$ T-solve(**B2T**(μ^P))
      if T-res = SAT **then return** SAT
      else $\phi^P \leftarrow \phi^P \wedge \neg\mu^P$

The *theory conflict clause* $\neg\mu^P$ is too weak; it blocks one assignment at a time. What is a better clause?

- $\phi \leftarrow (x = 1) \wedge ((x = 2) \vee (x = 3))$
- $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3)$
- $\mu^P \leftarrow b_1 \wedge b_2 \wedge b_3$
  - $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_2 \vee \neg b_3)$
- $\mu^P \leftarrow b_1 \wedge \neg b_2 \wedge b_3$
  - $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_2 \vee \neg b_3) \wedge (\neg b_1 \vee b_2 \vee \neg b_3)$
- $\mu^P \leftarrow b_1 \wedge b_2 \wedge \neg b_3$
  - $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_2 \vee \neg b_3) \wedge (\neg b_1 \vee b_2 \vee \neg b_3) \wedge (\neg b_1 \vee \neg b_2 \vee b_3)$
- UNSAT

Offline DPLL(T): improvements

Offline-DPLL$_T$(T-formula ϕ)
  $\phi^P \leftarrow$ **T2B**(ϕ)
  while (TRUE) **do**
    μ^P, res $\leftarrow$ **CDCL**(ϕ^P)
    if res = UNSAT **then return** UNSAT
    else
      T-res $\leftarrow$ T-solve(**B2T**(μ^P))
      if T-res = SAT **then return** SAT
      else $\phi^P \leftarrow \phi^P \wedge \neg\mu^P$

The *theory conflict clause* $\neg\mu^P$ is too weak; it blocks one assignment at a time. What is a better clause?

$\neg$T2B(MinimalUnsatCore(B2T(μ^P)))

- $\phi \leftarrow (x = 1) \wedge ((x = 2) \vee (x = 3))$
- $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3)$
- $\mu^P \leftarrow b_1 \wedge b_2 \wedge b_3$
  - $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_2 \vee \neg b_3)$
- $\mu^P \leftarrow b_1 \wedge \neg b_2 \wedge b_3$
  - $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_2 \vee \neg b_3) \wedge (\neg b_1 \vee b_2 \vee \neg b_3)$
- $\mu^P \leftarrow b_1 \wedge b_2 \wedge \neg b_3$
  - $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_2 \vee \neg b_3) \wedge (\neg b_1 \vee b_2 \vee \neg b_3) \wedge (\neg b_1 \vee \neg b_2 \vee b_3)$
- UNSAT

Offline DPLL(T): improvements

Offline-DPLL$_T$(T-formula ϕ)
  $\phi^P \leftarrow$ **T2B**(ϕ)
  while (TRUE) **do**
    μ^P, res $\leftarrow$ **CDCL**(ϕ^P)
    if res = UNSAT **then return** UNSAT
    else
      T-res $\leftarrow$ T-solve(**B2T**(μ^P))
      if T-res = SAT **then return** SAT
      else
        t $\leftarrow$ T2B(MINUNSATCORE(B2T(μ^P)))
        $\phi^P \leftarrow \phi^P \wedge \neg t$

- $\phi \leftarrow (x = 1) \wedge ((x = 2) \vee (x = 3))$
- $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3)$
- $\mu^P \leftarrow b_1 \wedge b_2 \wedge b_3$
  - $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_2)$
- $\mu^P \leftarrow b_1 \wedge \neg b_2 \wedge b_3$
  - $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_2) \wedge (\neg b_1 \vee \neg b_3)$
- UNSAT

Offline DPLL(T): improvements

Offline-DPLL$_T$(T-formula ϕ)

  $\phi^P \leftarrow$ **T2B**(ϕ)

  while (TRUE) **do**

    μ^P, res $\leftarrow$ **CDCL**(ϕ^P)

    if res = UNSAT **then return** UNSAT

    else

      T-res $\leftarrow$ T-solve(**B2T**(μ^P))

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

      else

        t $\leftarrow$ T2B(MinUnsatCore(B2T(μ^P)))

        $\phi^P \leftarrow \phi^P \wedge \neg t$

- $\phi \leftarrow (x = 1) \wedge ((x = 2) \vee (x = 3))$
- $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3)$
- $\mu^P \leftarrow b_1 \wedge b_2 \wedge b_3$
  - $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_2)$
- $\mu^P \leftarrow b_1 \wedge \neg b_2 \wedge b_3$
  - $\phi^P \leftarrow b_1 \wedge (b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_2) \wedge (\neg b_1 \vee \neg b_3)$
- UNSAT

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

Online DPLL(T) address this issue.

Online DPLL(T)

```
Online-DPLL_T(T-formula φ, T-assignment μ)
 if T-PREPROCESS(φ, μ) = CONFLICT then
   return UNSAT
 φP , μP ← T2B(φ), T2B(μ)
 while (TRUE) do
   T-DECIDE(φP, μP)
   while (TRUE) do
     res ← T-DEDUCE(φP, μP)
     if res = SAT then return SAT
     else if res = CONFLICT
       blevel ← T-ANALZECONFLICT(φP, μP)
       if (blevel < 0) then return UNSAT
       else T-BACKTRACK(blevel, φP, μP)
     else break
```

Online DPLL(T)

```
Online-DPLL_T(T-formula φ, T-assignment μ)
 if T-PREPROCESS(φ, μ) = CONFLICT then
   return UNSAT
 φP , μP ← T2B(φ), T2B(μ)
 while (TRUE) do
   T-DECIDE(φP, μP)
   while (TRUE) do
     res ← T-DEDUCE(φP, μP)
     if res = SAT then return SAT
     else if res = CONFLICT
       blevel ← T-ANALZECONFLICT(φP, μP)
       if (blevel < 0) then return UNSAT
       else T-BACKTRACK(blevel, φP, μP)
     else break
```

Everything passed by reference.

All procedures have access to T2B and B2T.

Online DPLL(T): T-PREPROCESS

```
Online-DPLL_T(T-formula φ, T-assignment μ)
  if T-PREPROCESS(φ, μ) = CONFLICT then
    return UNSAT
  φP , μP ← T2B(φ), T2B(μ)
  while (TRUE) do
    T-DECIDE(φP, μP)
    while (TRUE) do
      res ← T-DEDUCE(φP, μP)
      if res = SAT then return SAT
      else if res = CONFLICT
        blevel ← T-ANALZECONFLICT(φP, μP)
        if (blevel < 0) then return UNSAT
        else T-BACKTRACK(blevel, φP, μP)
      else break
```

Simplifies φ and updates μ, if needed, so that equisatisfiability is preserved.

Common simplifications:

- Drop dual operators
- Exploit associativity
- Sort arguments
- Exploit theory-specific properties

Online DPLL(T): T-PREPROCESS

```
Online-DPLL_T(T-formula φ, T-assignment μ)
  if T-PREPROCESS(φ, μ) = CONFLICT then
    return UNSAT
  φP , μP ← T2B(φ), T2B(μ)
  while (TRUE) do
    T-DECIDE(φP, μP)
    while (TRUE) do
      res ← T-DEDUCE(φP, μP)
      if res = SAT then return SAT
      else if res = CONFLICT
        blevel ← T-ANALZECONFLICT(φP, μP)
        if (blevel < 0) then return UNSAT
        else T-BACKTRACK(blevel, φP, μP)
      else break
```

Simplifies φ and updates μ, if needed, so that equisatisfiability is preserved.

Common simplifications:

- Drop dual operators
- Exploit associativity
- Sort arguments
- Exploit theory-specific properties

Online DPLL(T): T-DECIDE

```
Online-DPLL_T(T-formula φ, T-assignment μ)
  if T-PREPROCESS(φ, μ) = CONFLICT then
    return UNSAT
  φP , μP ← T2B(φ), T2B(μ)
  while (TRUE) do
    T-DECIDE(φP, μP)
    while (TRUE) do
      res ← T-DEDUCE(φP, μP)
      if res = SAT then return SAT
      else if res = CONFLICT
        blevel ← T-ANALZECONFLICT(φP, μP)
        if (blevel < 0) then return UNSAT
        else T-BACKTRACK(blevel, φP, μP)
      else break
```

Analogous to DECIDE in CDCL:

- Selects an unassigned l^P literal and adds it to μ^P.
- May consider the semantics of literals in T.

Online DPLL(T): T-DECIDE

```
Online-DPLL_T(T-formula φ, T-assignment μ)
  if T-PREPROCESS(φ, μ) = CONFLICT then
    return UNSAT
  φ^P , μ^P ← T2B(φ), T2B(μ)
  while (TRUE) do
    T-DECIDE(φ^P, μ^P)
    while (TRUE) do
      res ← T-DEDUCE(φ^P, μ^P)
      if res = SAT then return SAT
      else if res = CONFLICT
        blevel ← T-ANALZECONFLICT(φ^P, μ^P)
        if (blevel < 0) then return UNSAT
        else T-BACKTRACK(blevel, φ^P, μ^P)
      else break
```

Analogous to DECIDE in CDCL:

- Selects an unassigned l^P literal and adds it to μ^P.
- May consider the semantics of literals in T.

Online DPLL(T): T-DEDUCE

```
Online-DPLL_T(T-formula φ, T-assignment μ)
  if T-PREPROCESS(φ, μ) = CONFLICT then
    return UNSAT
  φP , μP ← T2B(φ), T2B(μ)
  while (TRUE) do
    T-DECIDE(φP, μP)
    while (TRUE) do
      res ← T-DEDUCE(φP, μP)
      if res = SAT then return SAT
      else if res = CONFLICT
        blevel ← T-ANALZECONFLICT(φP, μP)
        if (blevel < 0) then return UNSAT
        else T-BACKTRACK(blevel, φP, μP)
      else break
```

Applies BCP to ϕ^P and μ^P until

- μ^P propositionally violates ϕ^P: returns CONFLICT.
- μ^P propositionally satisfies ϕ^P: invokes T-solver on B2T(μ^P) and returns SAT if T-solver does. Otherwise returns CONFLICT.
- no more literals can be deduced: invokes T-solver on partial assignment B2T(μ^P) and returns CONFLICT is T-solver returns UNSAT. This is *early pruning*. May also do *theory propagation*.

Online DPLL(T): T-DEDUCE

```
Online-DPLL_T(T-formula φ, T-assignment μ)
  if T-PREPROCESS(φ, μ) = CONFLICT then
    return UNSAT
  φP , μP ← T2B(φ), T2B(μ)
  while (TRUE) do
    T-DECIDE(φP, μP)
    while (TRUE) do
      res ← T-DEDUCE(φP, μP)
      if res = SAT then return SAT
      else if res = CONFLICT
        blevel ← T-ANALZECONFLICT(φP, μP)
        if (blevel < 0) then return UNSAT
        else T-BACKTRACK(blevel, φP, μP)
      else break
```

Applies BCP to ϕ^P and μ^P until

- μ^P propositionally violates ϕ^P: returns CONFLICT.
- μ^P propositionally satisfies ϕ^P: invokes T-solver on B2T(μ^P) and returns SAT if T-solver does. Otherwise returns CONFLICT.
- no more literals can be deduced: invokes T-solver on partial assignment B2T(μ^P) and returns CONFLICT is T-solver returns UNSAT. This is *early pruning*. May also do *theory propagation*.

Online DPLL(T): T-DEDUCE

```
Online-DPLL_T(T-formula φ, T-assignment μ)
  if T-PREPROCESS(φ, μ) = CONFLICT then
    return UNSAT
  φP , μP ← T2B(φ), T2B(μ)
  while (TRUE) do
    T-DECIDE(φP, μP)
    while (TRUE) do
      res ← T-DEDUCE(φP, μP)
      if res = SAT then return SAT
      else if res = CONFLICT
        blevel ← T-ANALZECONFLICT(φP, μP)
        if (blevel < 0) then return UNSAT
        else T-BACKTRACK(blevel, φP, μP)
      else break
```

Applies BCP to ϕ^P and μ^P until

- μ^P propositionally violates ϕ^P: returns CONFLICT.
- μ^P propositionally satisfies ϕ^P: invokes T-solver on B2T(μ^P) and returns SAT if T-solver does. Otherwise returns CONFLICT.
- no more literals can be deduced: invokes T-solver on partial assignment B2T(μ^P) and returns CONFLICT is T-solver returns UNSAT. This is *early pruning*. May also do *theory propagation*.

Online DPLL(T): T-DEDUCE

```
Online-DPLL_T(T-formula φ, T-assignment μ)
  if T-PREPROCESS(φ, μ) = CONFLICT then
    return UNSAT
  φP , μP ← T2B(φ), T2B(μ)
  while (TRUE) do
    T-DECIDE(φP, μP)
    while (TRUE) do
      res ← T-DEDUCE(φP, μP)
      if res = SAT then return SAT
      else if res = CONFLICT
        blevel ← T-ANALZECONFLICT(φP, μP)
        if (blevel < 0) then return UNSAT
        else T-BACKTRACK(blevel, φP, μP)
      else break
```

Applies BCP to ϕ^P and μ^P until

- μ^P propositionally violates ϕ^P: returns CONFLICT.
- μ^P propositionally satisfies ϕ^P: invokes T-solver on B2T(μ^P) and returns SAT if T-solver does. Otherwise returns CONFLICT.
- no more literals can be deduced: invokes T-solver on partial assignment B2T(μ^P) and returns CONFLICT is T-solver returns UNSAT. This is *early pruning*. May also do *theory propagation*.

Online DPLL(T): T-ANALYZECONFLICT

```
Online-DPLL_T(T-formula φ, T-assignment μ)
 if T-PREPROCESS(φ, μ) = CONFLICT then
   return UNSAT
 φP , μP ← T2B(φ), T2B(μ)
 while (TRUE) do
   T-DECIDE(φP, μP)
   while (TRUE) do
     res ← T-DEDUCE(φP, μP)
     if res = SAT then return SAT
     else if res = CONFLICT
       blevel ← T-ANALZECONFLICT(φP, μP)
       if (blevel < 0) then return UNSAT
       else T-BACKTRACK(blevel, φP, μP)
     else break
```

Extends ANALYZECONFLICT from CDCL:

- if the conflict is caused by a boolean (BCP) failure, returns the same blevel and conflict clause as ANALYZECONFLICT
- if the conflict is caused by a theory failure, returns a mixed boolean+theory conflict clause

Online DPLL(T): T-AnalyzeConflict

```
Online-DPLL_T(T-formula φ, T-assignment μ)
 if T-PREPROCESS(φ, μ) = CONFLICT then
   return UNSAT
 φP , μP ← T2B(φ), T2B(μ)
 while (TRUE) do
   T-DECIDE(φP, μP)
   while (TRUE) do
     res ← T-DEDUCE(φP, μP)
     if res = SAT then return SAT
     else if res = CONFLICT
       blevel ← T-ANALZECONFLICT(φP, μP)
       if (blevel < 0) then return UNSAT
       else T-BACKTRACK(blevel, φP, μP)
     else break
```

Extends AnalyzeConflict from CDCL:

- if the conflict is caused by a boolean (BCP) failure, returns the same blevel and conflict clause as AnalyzeConflict
- if the conflict is caused by a theory failure, returns a mixed boolean+theory conflict clause

Online DPLL(T): T-ANALYZECONFLICT

```
Online-DPLL_T(T-formula φ, T-assignment μ)
  if T-PREPROCESS(φ, μ) = CONFLICT then
    return UNSAT
  φP , μP ← T2B(φ), T2B(μ)
  while (TRUE) do
    T-DECIDE(φP, μP)
    while (TRUE) do
      res ← T-DEDUCE(φP, μP)
      if res = SAT then return SAT
      else if res = CONFLICT
        blevel ← T-ANALZECONFLICT(φP, μP)
        if (blevel < 0) then return UNSAT
        else T-BACKTRACK(blevel, φP, μP)
      else break
```

Extends ANALYZECONFLICT from CDCL:

- if the conflict is caused by a boolean (BCP) failure, returns the same blevel and conflict clause as ANALYZECONFLICT
- if the conflict is caused by a theory failure, returns a mixed boolean+theory conflict clause

Online DPLL(T): T-BACKTRACK

```
Online-DPLL_T(T-formula φ, T-assignment μ)
 if T-PREPROCESS(φ, μ) = CONFLICT then
   return UNSAT
 φP , μP ← T2B(φ), T2B(μ)
 while (TRUE) do
   T-DECIDE(φP, μP)
   while (TRUE) do
     res ← T-DEDUCE(φP, μP)
     if res = SAT then return SAT
     else if res = CONFLICT
       blevel ← T-ANALZECONFLICT(φP, μP)
       if (blevel < 0) then return UNSAT
       else T-BACKTRACK(blevel, φP, μP)
     else break
```

Analogous to BACKTRACK in CDCL:

- adds learned clause to ϕ^P (*T-learning*).
- backtracks to blevel by undoing all the assignments > blevel (*T-backjumping*).

Online DPLL(T): T-BACKTRACK

```
Online-DPLL_T(T-formula φ, T-assignment μ)
 if T-PREPROCESS(φ, μ) = CONFLICT then
  return UNSAT
 φP , μP ← T2B(φ), T2B(μ)
 while (TRUE) do
  T-DECIDE(φP, μP)
  while (TRUE) do
   res ← T-DEDUCE(φP, μP)
   if res = SAT then return SAT
   else if res = CONFLICT
    blevel ← T-ANALZECONFLICT(φP, μP)
    if (blevel < 0) then return UNSAT
    else T-BACKTRACK(blevel, φP, μP)
   else break
```

Analogous to BACKTRACK in CDCL:

- adds learned clause to ϕ^P (*T-learning*).
- backtracks to blevel by undoing all the assignments > blevel (*T-backjumping*).

Online DPLL(T) example

T_R-formula ϕ and ϕ^p:

1. $\neg(2x_2 - x_3 > 2) \vee A_1$, $\neg b_1 \vee A_1$
2. $\neg A_2 \vee (x_1 - x_5 \leq 1)$, $\neg A_2 \vee b_2$
3. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $b_3 \vee A_2$
4. $\neg(2x_3 + x_4 \geq 5) \vee \neg(3x_1 - x_3 \leq 6) \vee \neg A_1$, $\neg b_4 \vee \neg b_5 \vee \neg A_1$
5. $A_1 \vee (3x_1 - 2x_2 \leq 3)$, $A_1 \vee b_3$
6. $(x_2 - x_4 \leq 6) \vee (x_5 = 5 - 3x_4) \vee \neg A_1$, $b_6 \vee b_7 \vee \neg A_1$
7. $A_1 \vee (x_3 = 3x_5 + 4) \vee A_2$, $A_1 \vee b_8 \vee A_2$

Online DPLL(T) example: T-Decide

T_R-formula ϕ and ϕ^p:

1. $\neg(2x_2 - x_3 > 2) \vee A_1$, $\neg b_1 \vee A_1$
2. $\neg A_2 \vee (x_1 - x_5 \leq 1)$, $\neg A_2 \vee b_2$
3. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $b_3 \vee A_2$
4. $\neg(2x_3 + x_4 \geq 5) \vee \neg(3x_1 - x_3 \leq 6) \vee \neg A_1$, $\neg b_4 \vee \neg b_5 \vee \neg A_1$
5. $A_1 \vee (3x_1 - 2x_2 \leq 3)$, $A_1 \vee b_3$
6. $(x_2 - x_4 \leq 6) \vee (x_5 = 5 - 3x_4) \vee \neg A_1$, $b_6 \vee b_7 \vee \neg A_1$
7. $A_1 \vee (x_3 = 3x_5 + 4) \vee A_2$, $A_1 \vee b_8 \vee A_2$

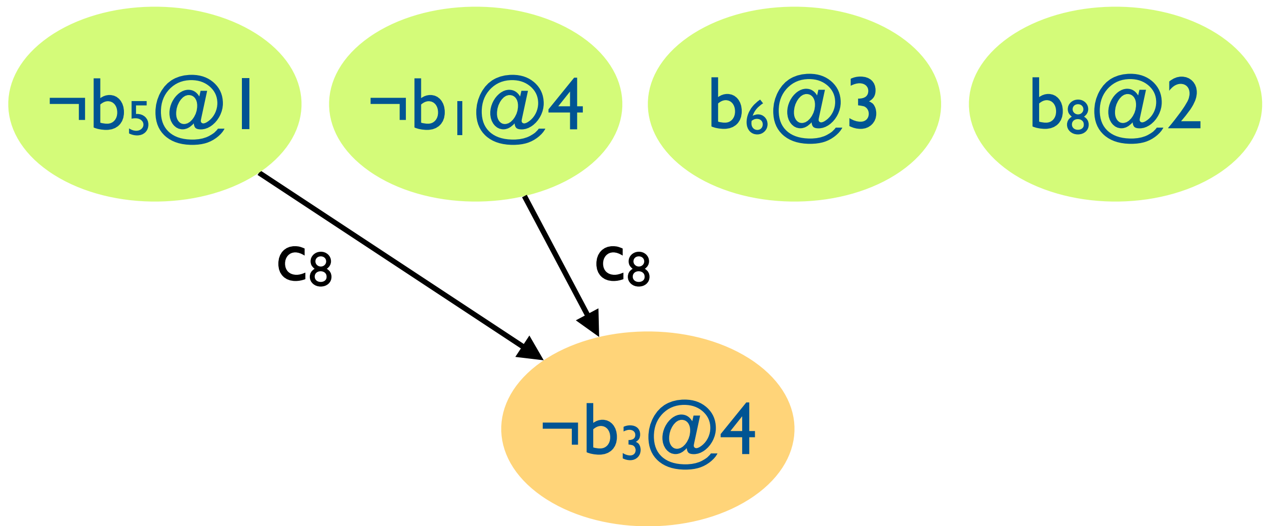
$\neg b_5@1$ $\neg b_1@4$ $b_6@3$ $b_8@2$

T-DECIDE makes 4 decisions.

Online DPLL(T) example: T-Deduce

T_R-formula ϕ and ϕ^p:

1. $\neg(2x_2 - x_3 > 2) \vee A_1$, $\neg b_1 \vee A_1$
2. $\neg A_2 \vee (x_1 - x_5 \leq 1)$, $\neg A_2 \vee b_2$
3. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $b_3 \vee A_2$
4. $\neg(2x_3 + x_4 \geq 5) \vee \neg(3x_1 - x_3 \leq 6) \vee \neg A_1$, $\neg b_4 \vee \neg b_5 \vee \neg A_1$
5. $A_1 \vee (3x_1 - 2x_2 \leq 3)$, $A_1 \vee b_3$
6. $(x_2 - x_4 \leq 6) \vee (x_5 = 5 - 3x_4) \vee \neg A_1$, $b_6 \vee b_7 \vee \neg A_1$
7. $A_1 \vee (x_3 = 3x_5 + 4) \vee A_2$, $A_1 \vee b_8 \vee A_2$
8. $b_5 \vee b_1 \vee \neg b_3$

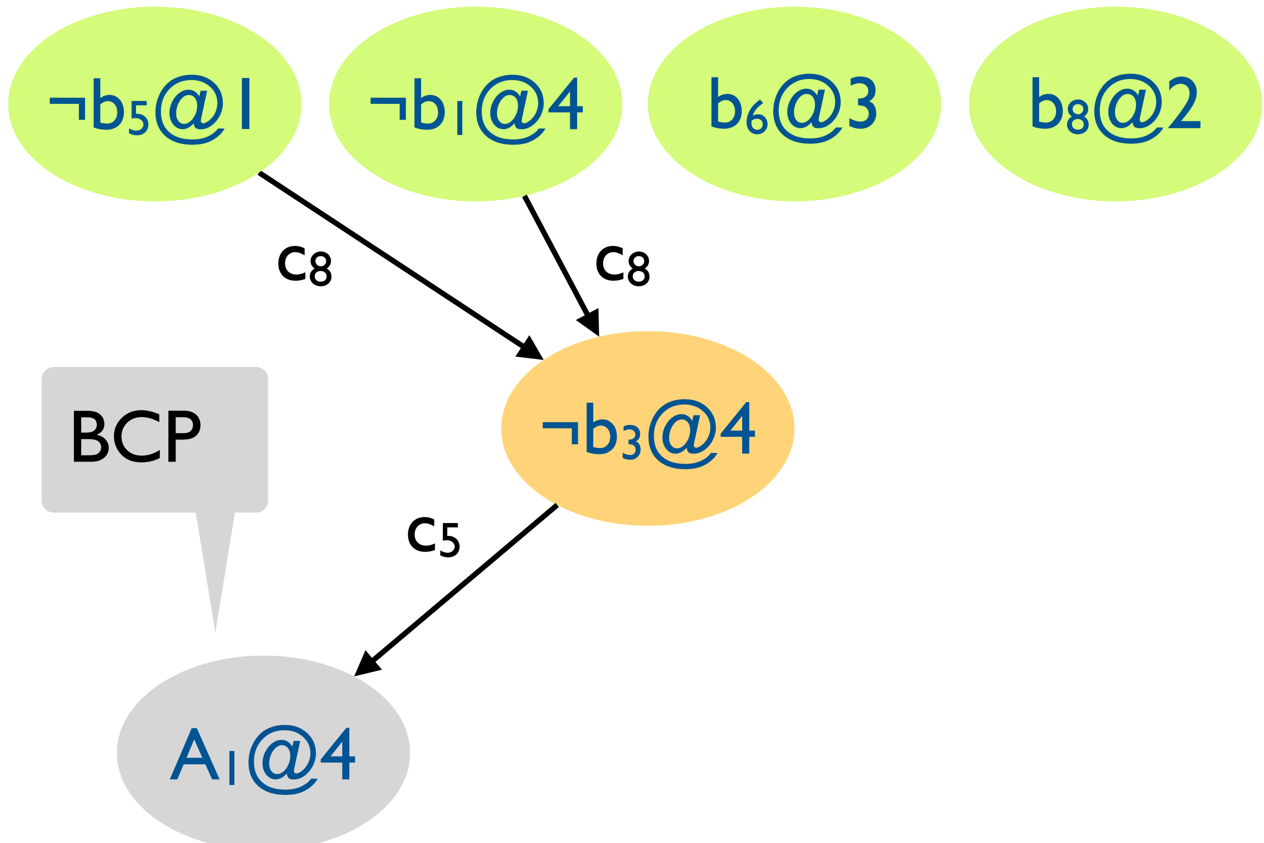


Early pruning & T-propagation.

Online DPLL(T) example: T-Deduce

T_R-formula ϕ and ϕ^p:

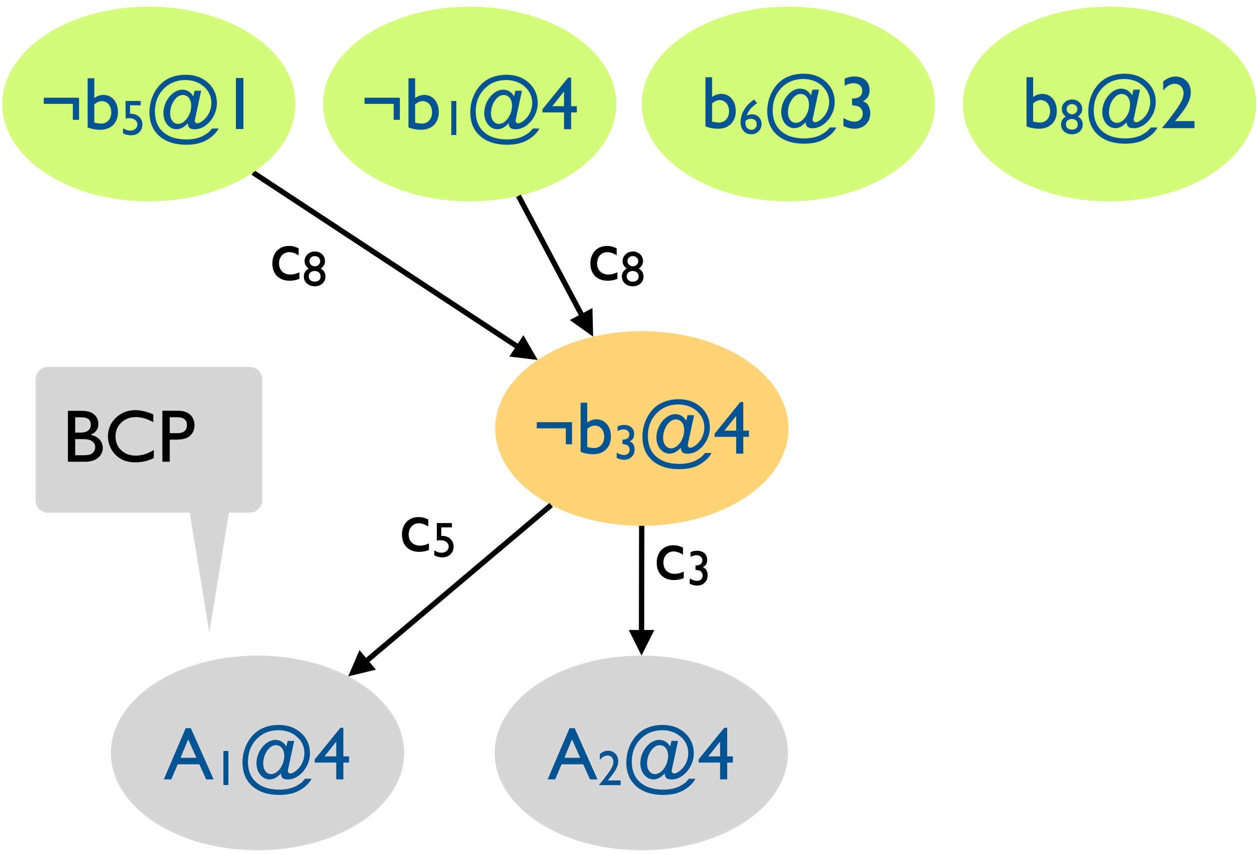
1. $\neg(2x_2 - x_3 > 2) \vee A_1$, $\neg b_1 \vee A_1$
2. $\neg A_2 \vee (x_1 - x_5 \leq 1)$, $\neg A_2 \vee b_2$
3. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $b_3 \vee A_2$
4. $\neg(2x_3 + x_4 \geq 5) \vee \neg(3x_1 - x_3 \leq 6) \vee \neg A_1$, $\neg b_4 \vee \neg b_5 \vee \neg A_1$
5. $A_1 \vee (3x_1 - 2x_2 \leq 3)$, $A_1 \vee b_3$
6. $(x_2 - x_4 \leq 6) \vee (x_5 = 5 - 3x_4) \vee \neg A_1$, $b_6 \vee b_7 \vee \neg A_1$
7. $A_1 \vee (x_3 = 3x_5 + 4) \vee A_2$, $A_1 \vee b_8 \vee A_2$
8. $b_5 \vee b_1 \vee \neg b_3$



Online DPLL(T) example: T-Deduce

T_R-formula ϕ and ϕ^P:

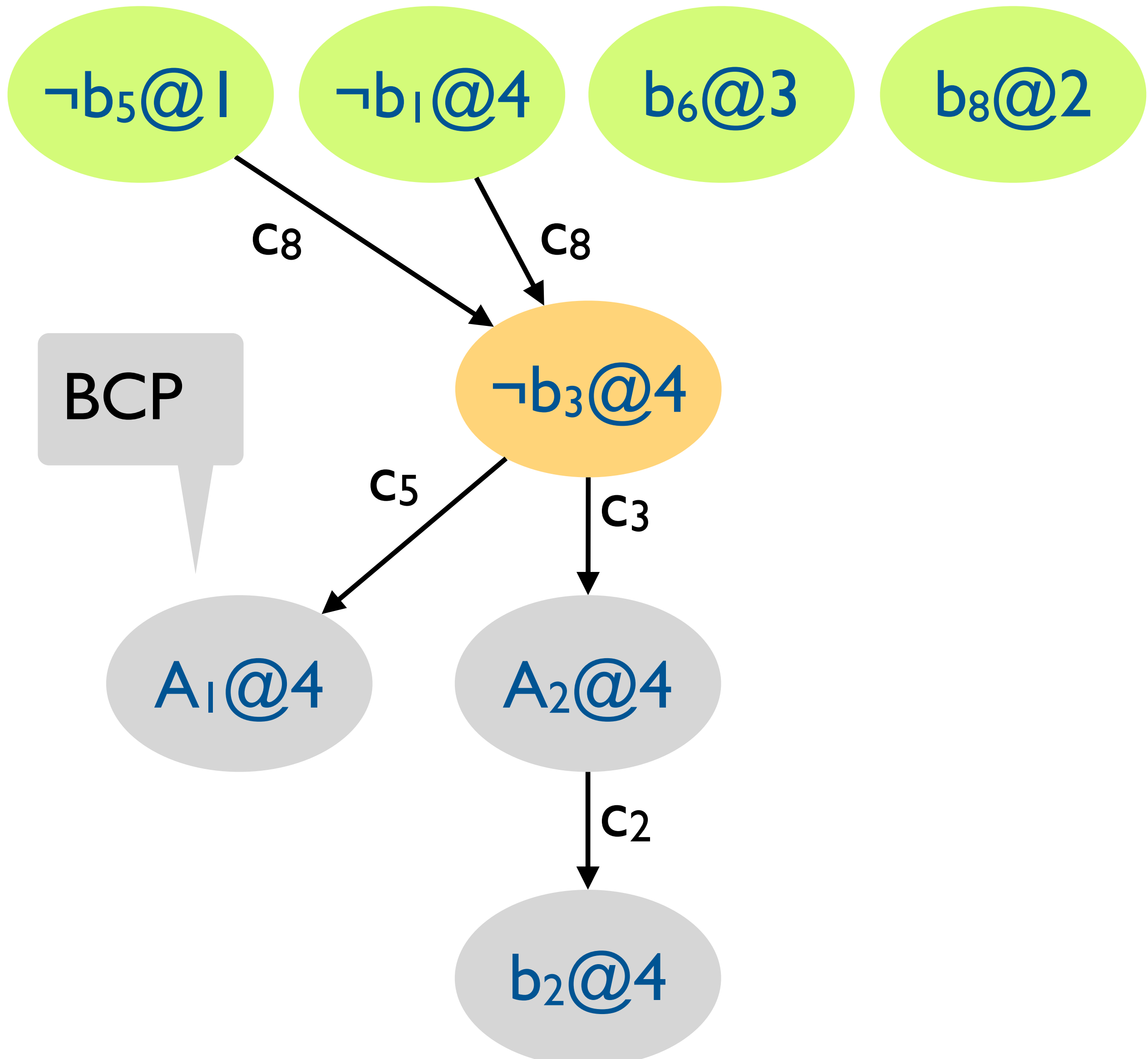
1. $\neg(2x_2 - x_3 > 2) \vee A_1$, $\neg b_1 \vee A_1$
2. $\neg A_2 \vee (x_1 - x_5 \leq 1)$, $\neg A_2 \vee b_2$
3. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $b_3 \vee A_2$
4. $\neg(2x_3 + x_4 \geq 5) \vee \neg(3x_1 - x_3 \leq 6) \vee \neg A_1$, $\neg b_4 \vee \neg b_5 \vee \neg A_1$
5. $A_1 \vee (3x_1 - 2x_2 \leq 3)$, $A_1 \vee b_3$
6. $(x_2 - x_4 \leq 6) \vee (x_5 = 5 - 3x_4) \vee \neg A_1$, $b_6 \vee b_7 \vee \neg A_1$
7. $A_1 \vee (x_3 = 3x_5 + 4) \vee A_2$, $A_1 \vee b_8 \vee A_2$
8. $b_5 \vee b_1 \vee \neg b_3$



Online DPLL(T) example: T-Deduce

T_R-formula ϕ and ϕ^p:

1. $\neg(2x_2 - x_3 > 2) \vee A_1$, $\neg b_1 \vee A_1$
2. $\neg A_2 \vee (x_1 - x_5 \leq 1)$, $\neg A_2 \vee b_2$
3. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $b_3 \vee A_2$
4. $\neg(2x_3 + x_4 \geq 5) \vee \neg(3x_1 - x_3 \leq 6) \vee \neg A_1$, $\neg b_4 \vee \neg b_5 \vee \neg A_1$
5. $A_1 \vee (3x_1 - 2x_2 \leq 3)$, $A_1 \vee b_3$
6. $(x_2 - x_4 \leq 6) \vee (x_5 = 5 - 3x_4) \vee \neg A_1$, $b_6 \vee b_7 \vee \neg A_1$
7. $A_1 \vee (x_3 = 3x_5 + 4) \vee A_2$, $A_1 \vee b_8 \vee A_2$
8. $b_5 \vee b_1 \vee \neg b_3$



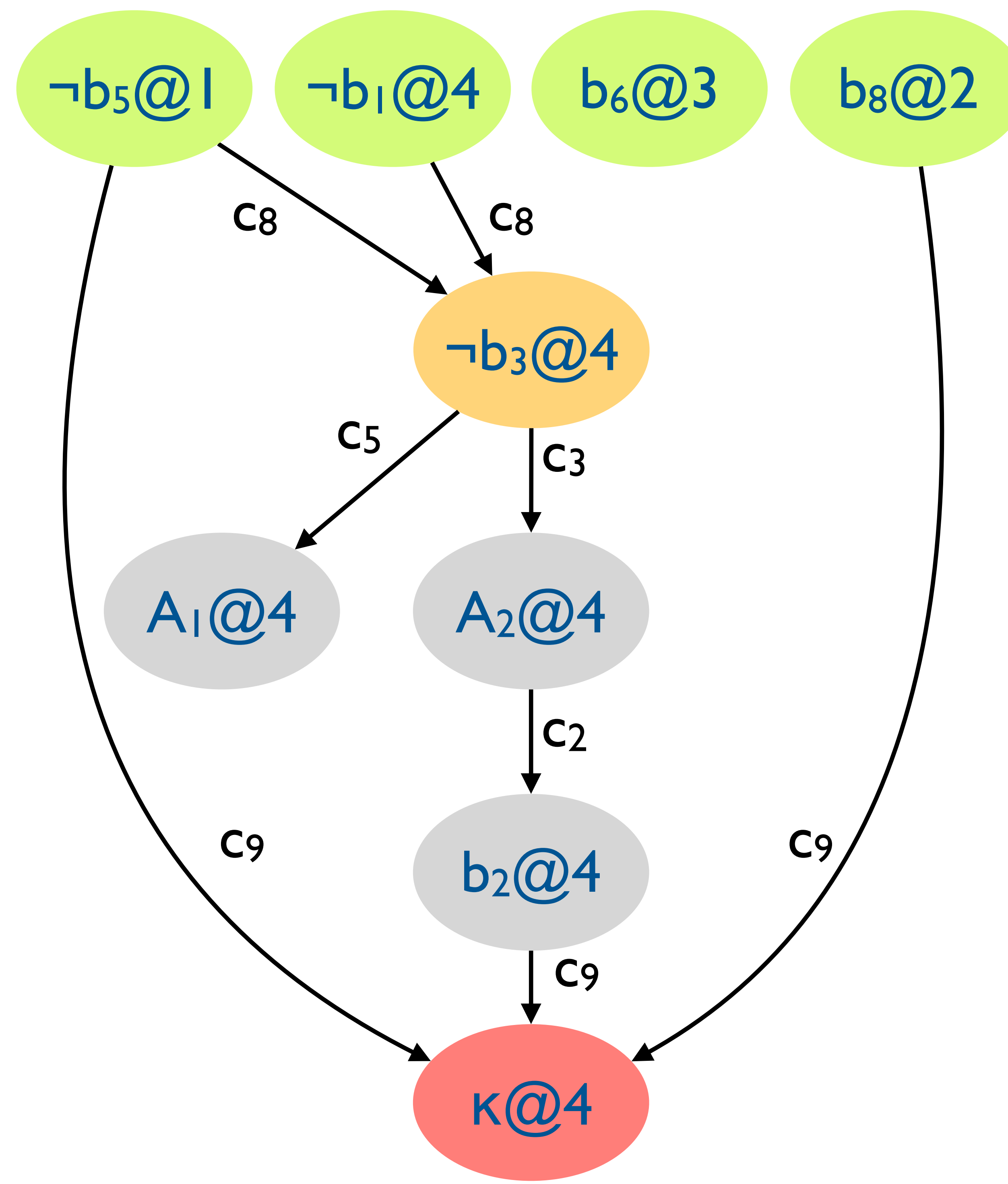
Online DPLL(T) example: T-Deduce

Current partial assignment is T-UNSAT.

T_R-formula ϕ and ϕ^p:

1. $\neg(2x_2 - x_3 > 2) \vee A_1$, $\neg b_1 \vee A_1$
2. $\neg A_2 \vee (x_1 - x_5 \leq 1)$, $\neg A_2 \vee b_2$
3. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $b_3 \vee A_2$
4. $\neg(2x_3 + x_4 \geq 5) \vee \neg(3x_1 - x_3 \leq 6) \vee \neg A_1$, $\neg b_4 \vee \neg b_5 \vee \neg A_1$
5. $A_1 \vee (3x_1 - 2x_2 \leq 3)$, $A_1 \vee b_3$
6. $(x_2 - x_4 \leq 6) \vee (x_5 = 5 - 3x_4) \vee \neg A_1$, $b_6 \vee b_7 \vee \neg A_1$
7. $A_1 \vee (x_3 = 3x_5 + 4) \vee A_2$, $A_1 \vee b_8 \vee A_2$
8. $b_5 \vee b_1 \vee \neg b_3$
9. $b_5 \vee \neg b_8 \vee \neg b_2$

Theory conflict.

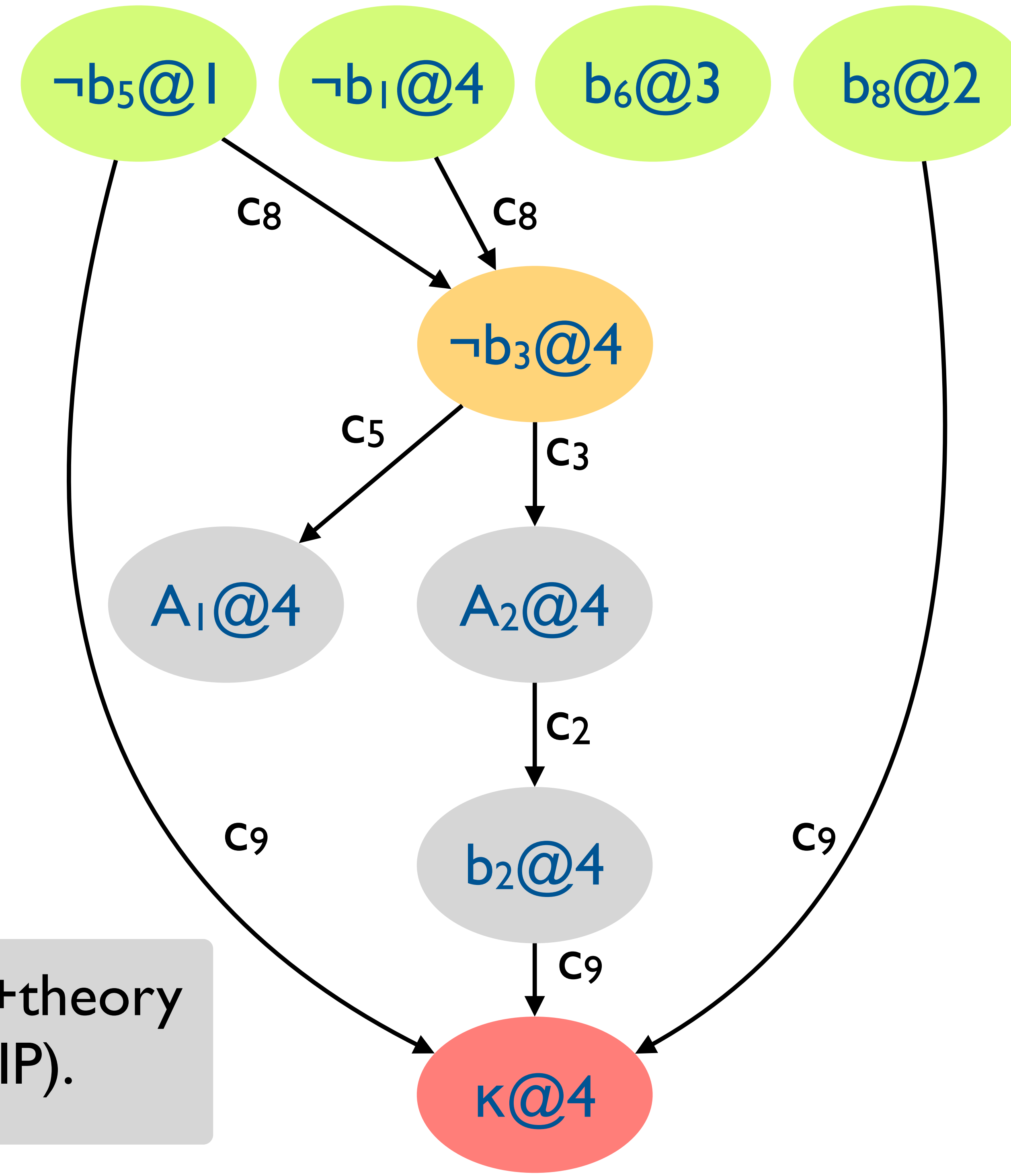


Online DPLL(T) example: T-AnalyzeConflict

T_R-formula ϕ and ϕ^p:

1. $\neg(2x_2 - x_3 > 2) \vee A_1$, $\neg b_1 \vee A_1$
2. $\neg A_2 \vee (x_1 - x_5 \leq 1)$, $\neg A_2 \vee b_2$
3. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $b_3 \vee A_2$
4. $\neg(2x_3 + x_4 \geq 5) \vee \neg(3x_1 - x_3 \leq 6) \vee \neg A_1$, $\neg b_4 \vee \neg b_5 \vee \neg A_1$
5. $A_1 \vee (3x_1 - 2x_2 \leq 3)$, $A_1 \vee b_3$
6. $(x_2 - x_4 \leq 6) \vee (x_5 = 5 - 3x_4) \vee \neg A_1$, $b_6 \vee b_7 \vee \neg A_1$
7. $A_1 \vee (x_3 = 3x_5 + 4) \vee A_2$, $A_1 \vee b_8 \vee A_2$
8. $b_5 \vee b_1 \vee \neg b_3$
9. $b_5 \vee \neg b_8 \vee \neg b_2$

Mixed boolean+theory conflict (first UIP).

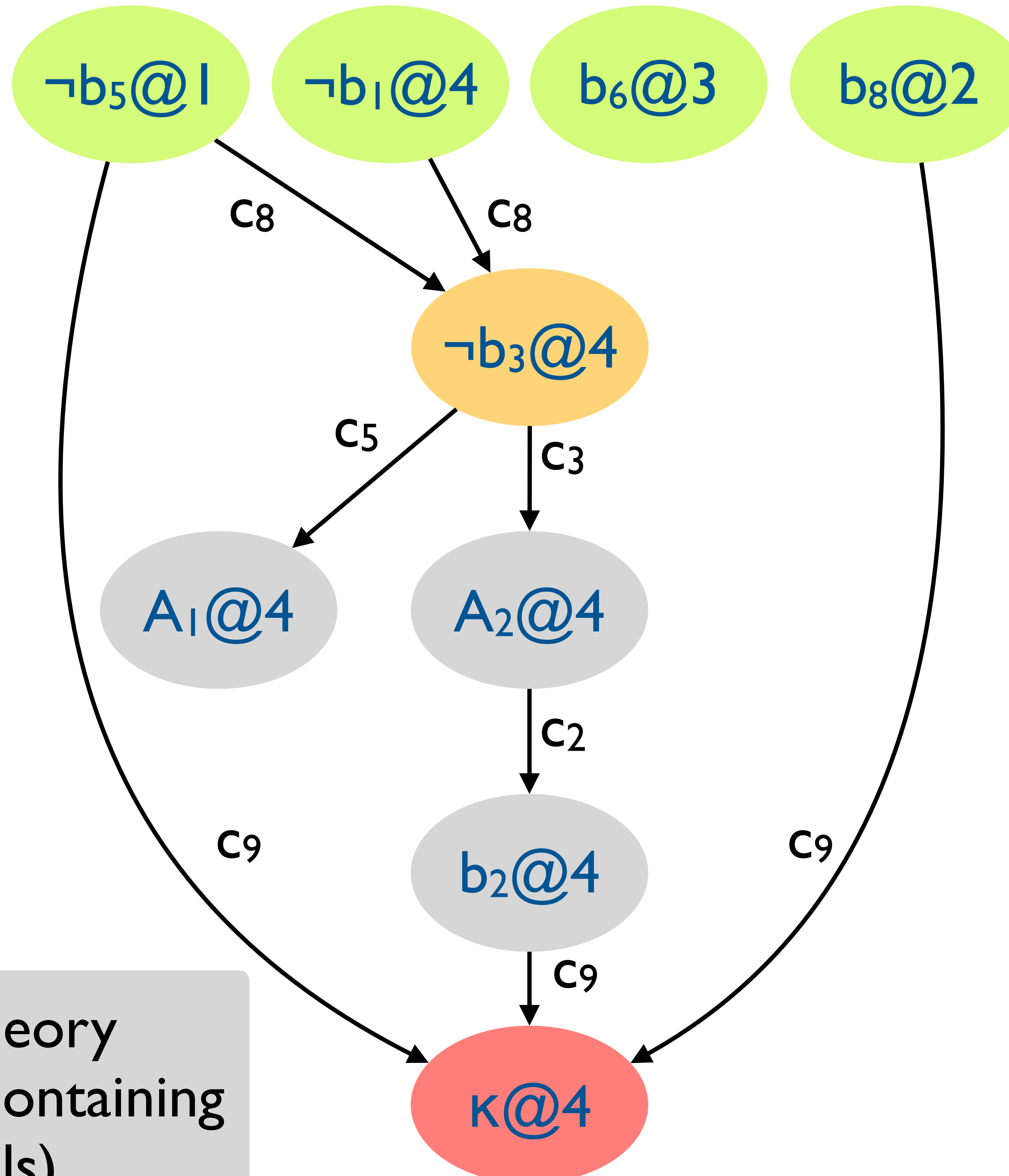


Online DPLL(T) example: T-AnalyzeConflict

T_R-formula ϕ and ϕ^p:

1. $\neg(2x_2 - x_3 > 2) \vee A_1$, $\neg b_1 \vee A_1$
2. $\neg A_2 \vee (x_1 - x_5 \leq 1)$, $\neg A_2 \vee b_2$
3. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $b_3 \vee A_2$
4. $\neg(2x_3 + x_4 \geq 5) \vee \neg(3x_1 - x_3 \leq 6) \vee \neg A_1$, $\neg b_4 \vee \neg b_5 \vee \neg A_1$
5. $A_1 \vee (3x_1 - 2x_2 \leq 3)$, $A_1 \vee b_3$
6. $(x_2 - x_4 \leq 6) \vee (x_5 = 5 - 3x_4) \vee \neg A_1$, $b_6 \vee b_7 \vee \neg A_1$
7. $A_1 \vee (x_3 = 3x_5 + 4) \vee A_2$, $A_1 \vee b_8 \vee A_2$
8. $b_5 \vee b_1 \vee \neg b_3$
9. $b_5 \vee \neg b_8 \vee \neg b_2$
10. $b_5 \vee \neg b_8 \vee b_1$

Mixed boolean+theory conflict (last UIP, containing just decision literals).



Online DPLL(T) example: T-Backtrack

T_R-formula ϕ and ϕ^p:

1. $\neg(2x_2 - x_3 > 2) \vee A_1$, $\neg b_1 \vee A_1$
2. $\neg A_2 \vee (x_1 - x_5 \leq 1)$, $\neg A_2 \vee b_2$
3. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $b_3 \vee A_2$
4. $\neg(2x_3 + x_4 \geq 5) \vee \neg(3x_1 - x_3 \leq 6) \vee \neg A_1$, $\neg b_4 \vee \neg b_5 \vee \neg A_1$
5. $A_1 \vee (3x_1 - 2x_2 \leq 3)$, $A_1 \vee b_3$
6. $(x_2 - x_4 \leq 6) \vee (x_5 = 5 - 3x_4) \vee \neg A_1$, $b_6 \vee b_7 \vee \neg A_1$
7. $A_1 \vee (x_3 = 3x_5 + 4) \vee A_2$, $A_1 \vee b_8 \vee A_2$
8. $b_5 \vee b_1 \vee \neg b_3$
9. $b_5 \vee \neg b_8 \vee \neg b_2$
10. $b_5 \vee \neg b_8 \vee b_1$

$\neg b_5 @ 1$

$b_8 @ 2$

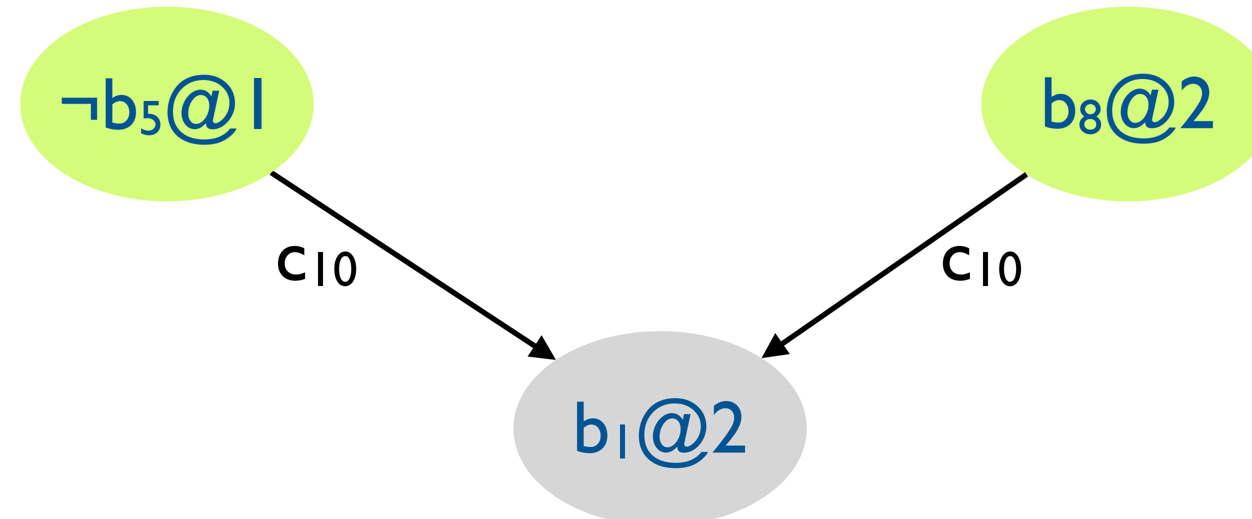
Backtrack to the second highest decision level (2).

Online DPLL(T) example: T-Backtrack

T_R-formula ϕ and ϕ^p:

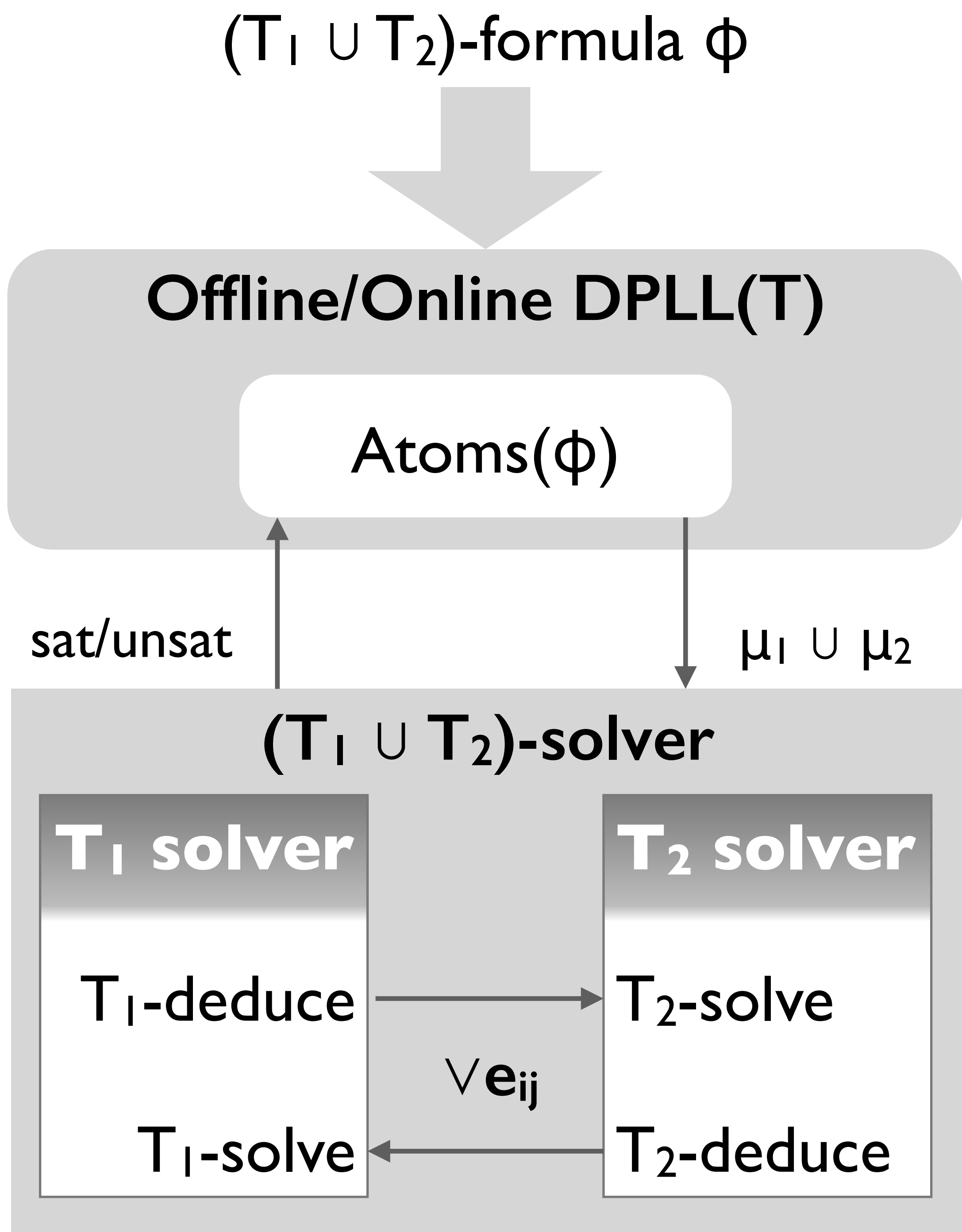
1. $\neg(2x_2 - x_3 > 2) \vee A_1$, $\neg b_1 \vee A_1$
2. $\neg A_2 \vee (x_1 - x_5 \leq 1)$, $\neg A_2 \vee b_2$
3. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $b_3 \vee A_2$
4. $\neg(2x_3 + x_4 \geq 5) \vee \neg(3x_1 - x_3 \leq 6) \vee \neg A_1$, $\neg b_4 \vee \neg b_5 \vee \neg A_1$
5. $A_1 \vee (3x_1 - 2x_2 \leq 3)$, $A_1 \vee b_3$
6. $(x_2 - x_4 \leq 6) \vee (x_5 = 5 - 3x_4) \vee \neg A_1$, $b_6 \vee b_7 \vee \neg A_1$
7. $A_1 \vee (x_3 = 3x_5 + 4) \vee A_2$, $A_1 \vee b_8 \vee A_2$
8. $b_5 \vee b_1 \vee \neg b_3$
9. $b_5 \vee \neg b_8 \vee \neg b_2$
10. $b_5 \vee \neg b_8 \vee b_1$

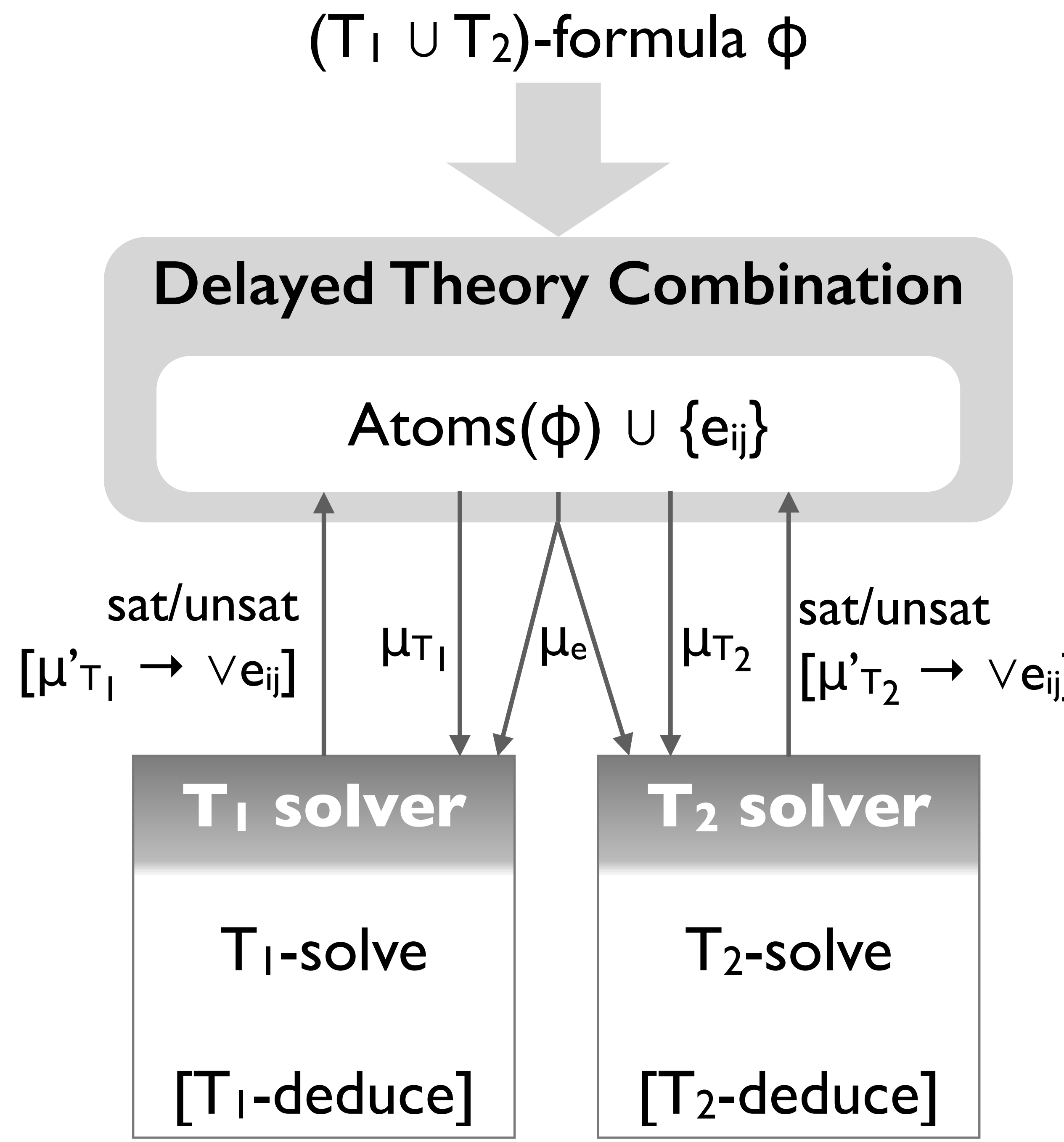
Backtrack to the second highest decision level (2).



b_1 immediately implied by BCP.

The DPLL(T) Framework





R. Bruttomesso et al., DTC vs NO, 2009

Delayed Theory Combination (DTC)

```
Online-DPLL_T(T-formula φ, T-assignment μ)
  if T-PREPROCESS(φ, μ) = CONFLICT then
    return UNSAT
  φ^P , μ^P ← T2B(φ), T2B(μ)
  while (TRUE) do
    T-DECIDE(φ^P, μ^P)
    while (TRUE) do
      res ← T-DEDUCE(φ^P, μ^P)
      if res = SAT then return SAT
      else if res = CONFLICT
        blevel ← T-ANALZECONFLICT(φ^P, μ^P)
        if (blevel < 0) then return UNSAT
        else T-BACKTRACK(blevel, φ^P, μ^P)
      else break
```

To get DTC, modify Online-DPLL$_T$ so that

- Truth values assigned to both atoms in ϕ and the *interface equalities* e_{ij} not in ϕ.
- T-DECIDE branches on interface equalities e_{ij} after μ propositionally satisfies ϕ.
- T-DEDUCE passes $\mu_i \cup \mu_e$ to each T_i-solver and returns SAT if both return SAT. Otherwise returns CONFLICT.
- T-ANALYZECONFLICT and T-BACKTRACK use the conflict clause (possibly containing interface equalities) from one of the T_i-solvers.
- Early pruning and T-PROPAGATION are performed.

Summary

Today

- The DPLL(T) framework for deciding SMT formulas

Next lecture

- Finite model finding: reasoning about quantified formulas over finite domains
- Last lecture on **computer-aided reasoning**
- It’s all **for software** afterwards!