Computer-Aided Reasoning for Software

CSE507

The DPLL(T) Framework

courses.cs.washington.edu/courses/cse507/14au/

Emina Torlak
emina@cs.washington.edu

Today

Today

Last lecture

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

Today

Last lecture

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

Today

- Deciding arbitrary boolean combinations of theory constraints

Today

Last lecture

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

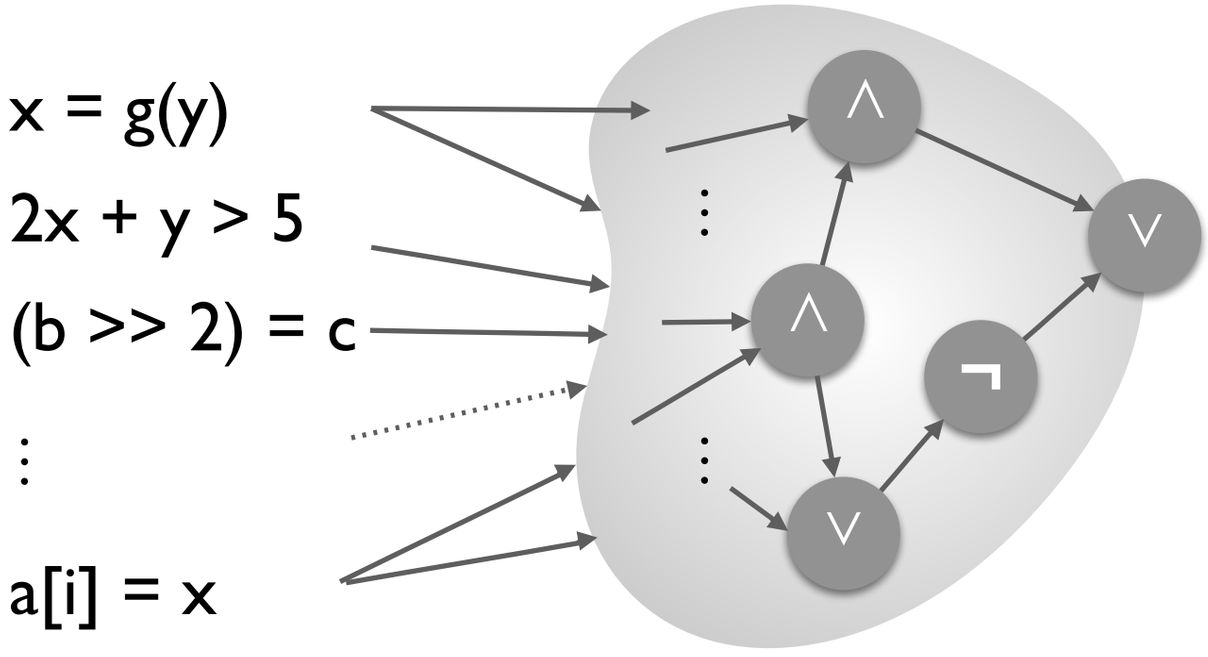
Today

- Deciding arbitrary boolean combinations of theory constraints

Reminders

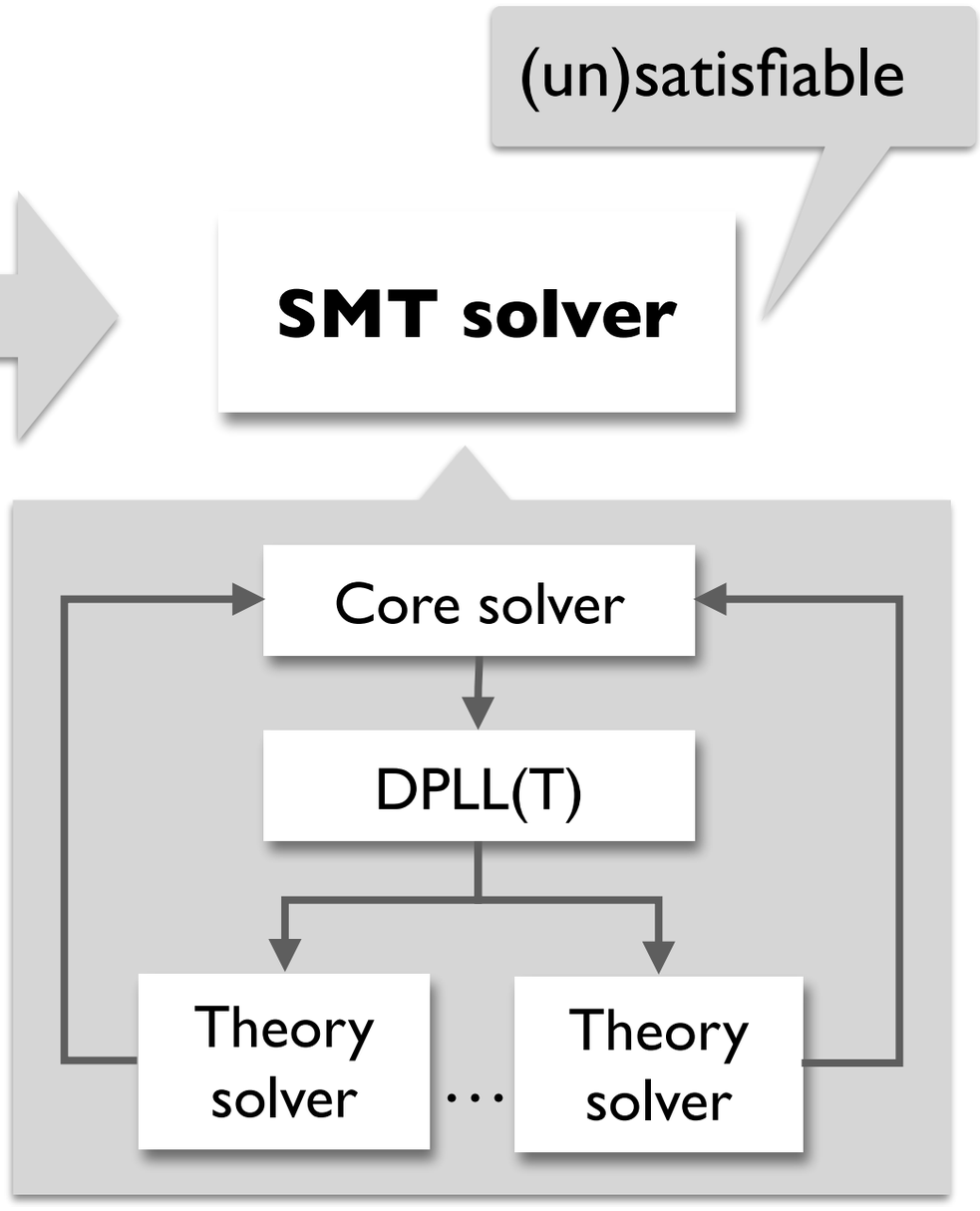
- Project proposals due at 11pm tonight
  - Submit via the **507 Dropbox** (one per team)
  - Follow the **formatting guidelines**

Satisfiability Modulo Theories (SMT)

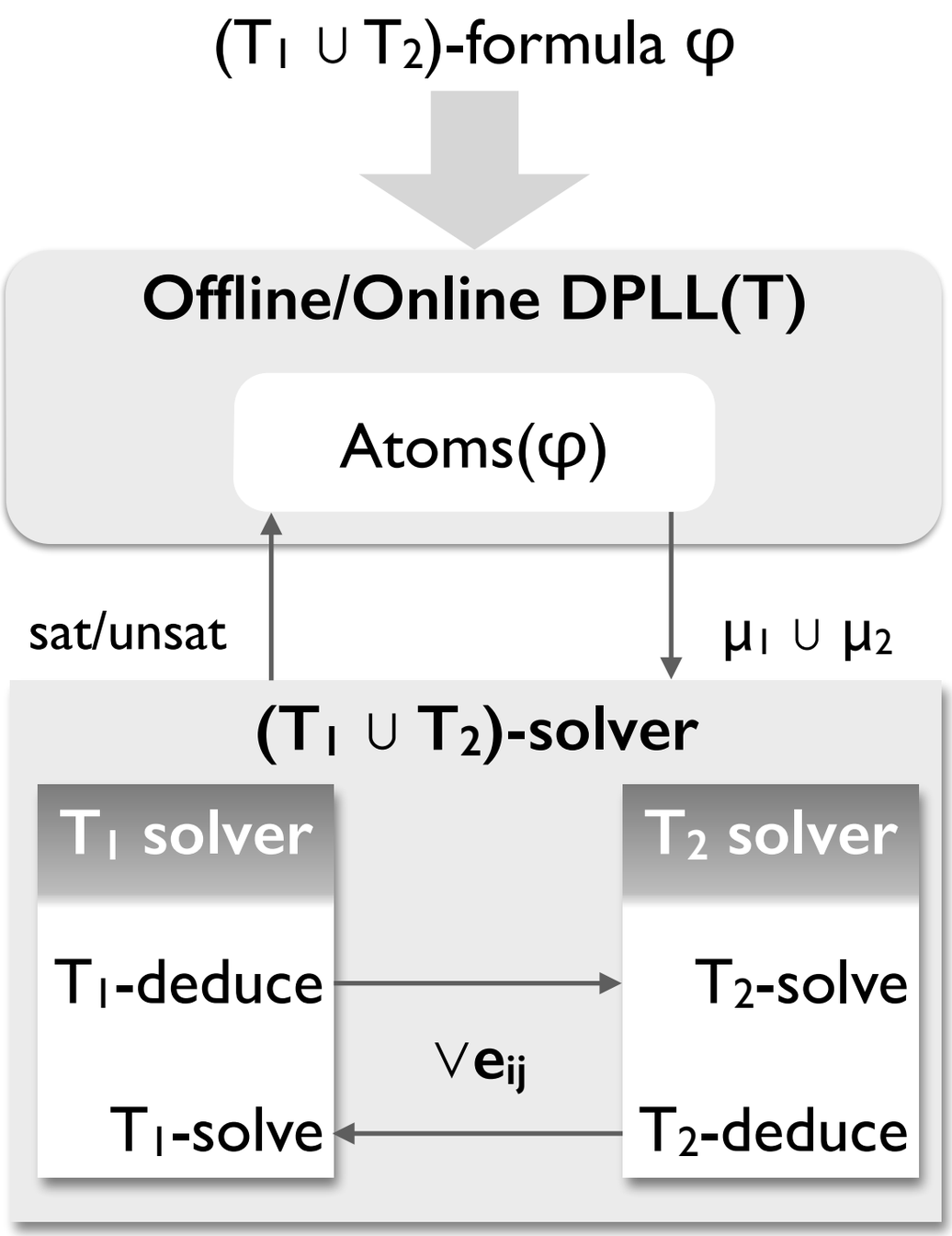


Theories

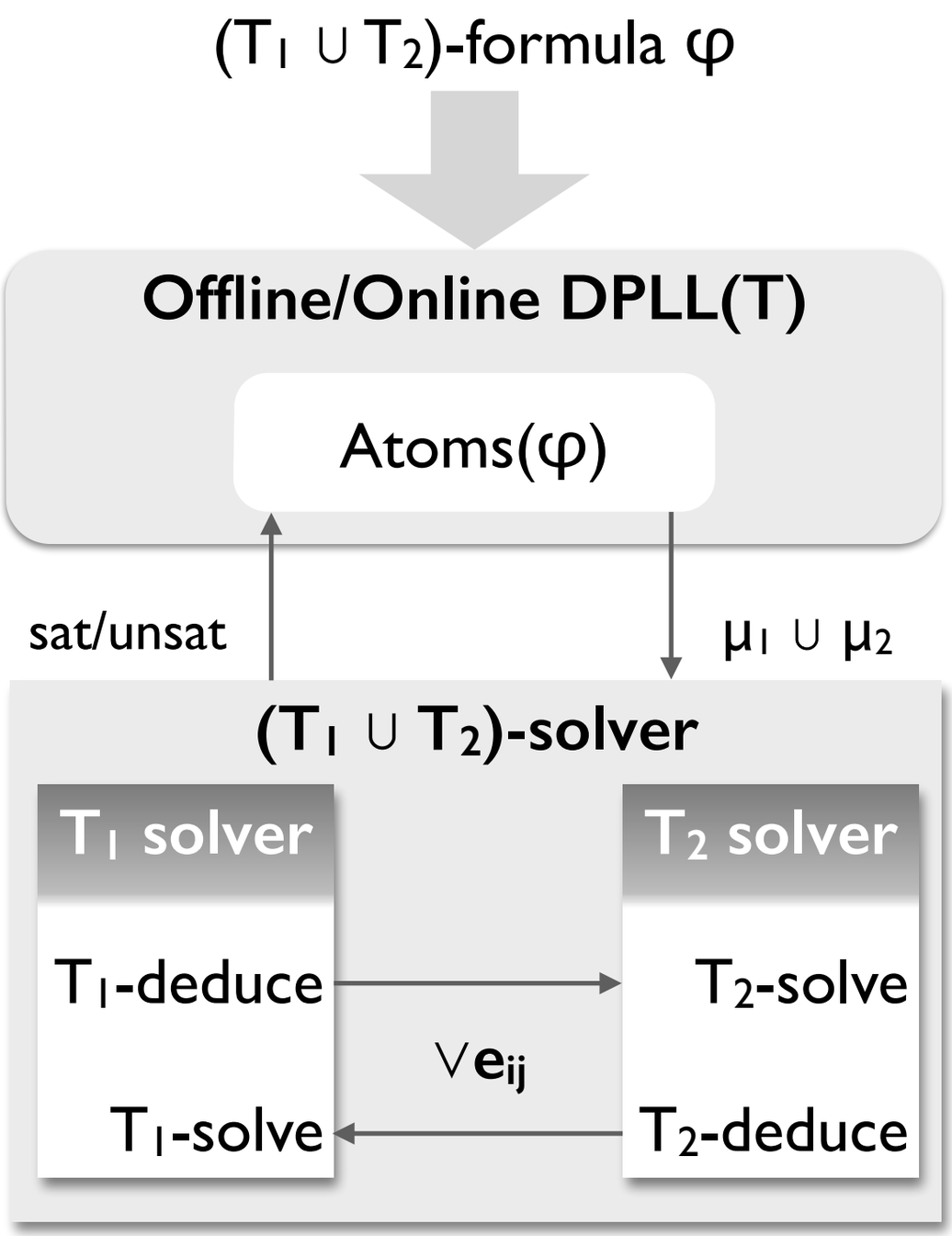
First-Order Logic

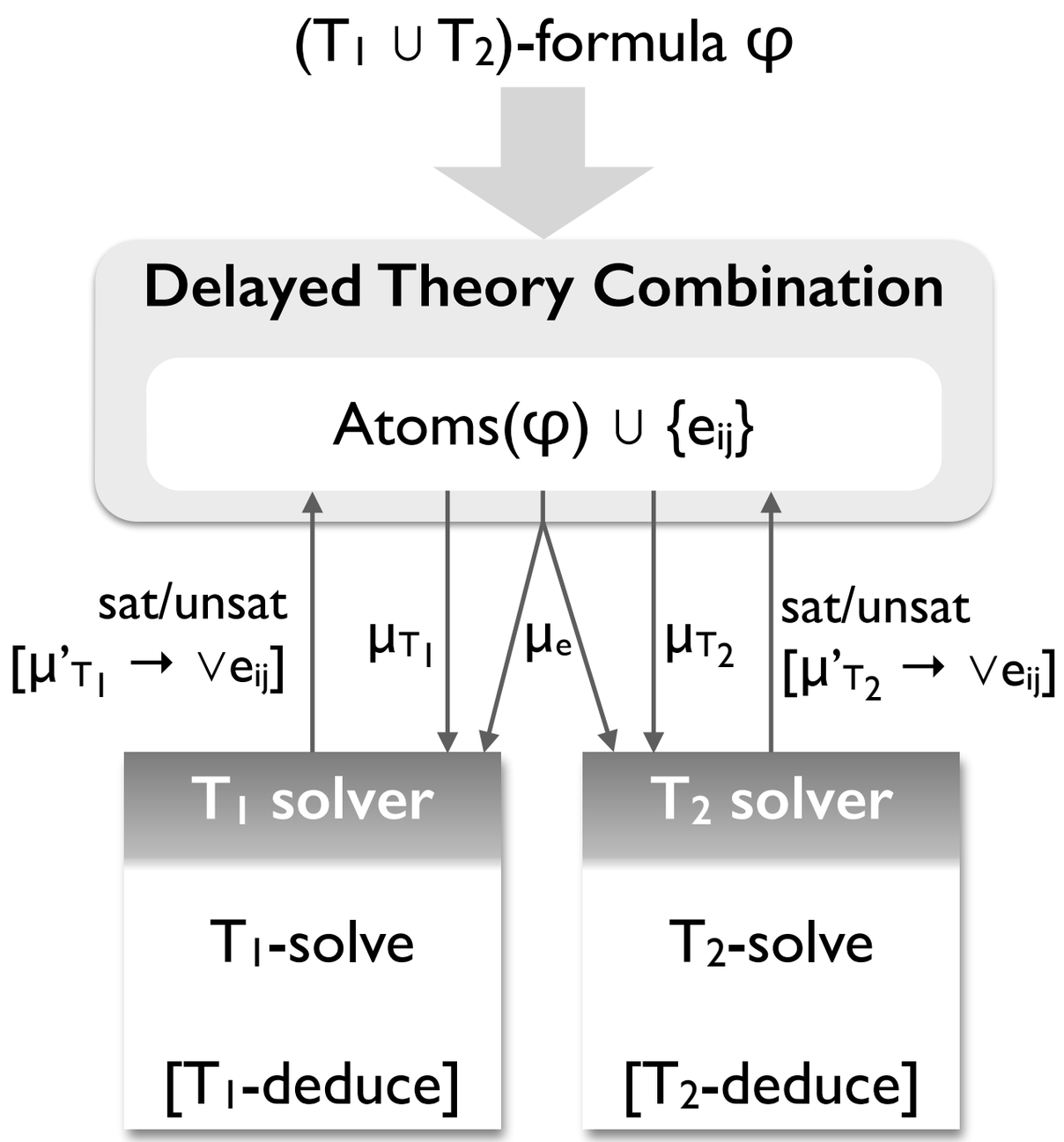


The DPLL(T) Framework



The DPLL(T) Framework





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

Offline DPLL(T)

```
Offline-DPLL_T(T-formula φ)
  φ^P ← T2B(φ)
  while (TRUE) do
    μ^P, res ← CDCL(φ^P)
    if res = UNSAT then return UNSAT
    else
      t, res ← T-solve(B2T(μ^P))
      if res = SAT then return SAT
      else φ^P ← φ^P ∧ T2B(t)
```

Offline DPLL(T)

```
Offline-DPLL_T(T-formula φ)
  φ^P ← T2B(φ)
  while (TRUE) do
    μ^P, res ← CDCL(φ^P)
    if res = UNSAT then return UNSAT
    else
      t, res ← T-solve(B2T(μ^P))
      if res = SAT then return SAT
      else φ^P ← φ^P ∧ T2B(t)
```

Assume φ is in CNF.

Offline DPLL(T)

```
Offline-DPLL_T(T-formula φ)
  φ^P ← T2B(φ)
  while (TRUE) do
    μ^P, res ← CDCL(φ^P)
    if res = UNSAT then return UNSAT
    else
      t, res ← T-solve(B2T(μ^P))
      if res = SAT then return SAT
      else φ^P ← φ^P ∧ T2B(t)
```

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)
  φ^P ← T2B(φ)
  while (TRUE) do
    μ^P, res ← CDCL(φ^P)
    if res = UNSAT then return UNSAT
    else
      t, res ← T-solve(B2T(μ^P))
      if res = SAT then return SAT
      else φ^P ← φ^P ∧ T2B(t)
```

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

If μ doesn't *propositionally satisfy* φ, T-solve returns a *theory conflict set*, whose negation is a *theory conflict clause* t. This clause blocks the current propositional assignment.

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 atom
- T2B(b_j) = b_j, if b_j is a boolean atom
- T2B($\varphi_1 \wedge \varphi_2$) = T2B($\varphi_1$) $\wedge$ T2B(φ_2)
- T2B($\varphi_1 \vee \varphi_2$) = T2B(φ_1) $\vee$ T2B(φ_2)
- T2B($\neg\varphi_1$) = $\neg$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 atom
- $T2B(b_j) = b_j$, if b_j is a boolean atom
- $T2B(\varphi_1 \wedge \varphi_2) = T2B(\varphi_1) \wedge T2B(\varphi_2)$
- $T2B(\varphi_1 \vee \varphi_2) = T2B(\varphi_1) \vee T2B(\varphi_2)$
- $T2B(\neg\varphi_1) = \neg T2B(\varphi_1)$

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

$B2T(\varphi^P) = T2B^{-1}(\varphi^P)$

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 atom
- $T2B(b_j) = b_j$, if b_j is a boolean atom
- $T2B(\varphi_1 \wedge \varphi_2) = T2B(\varphi_1) \wedge T2B(\varphi_2)$
- $T2B(\varphi_1 \vee \varphi_2) = T2B(\varphi_1) \vee T2B(\varphi_2)$
- $T2B(\neg\varphi_1) = \neg T2B(\varphi_1)$

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

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

$B2T(\varphi^P) = T2B^{-1}(\varphi^P)$

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 atom
- $T2B(b_j) = b_j$, if b_j is a boolean atom
- $T2B(\varphi_1 \wedge \varphi_2) = T2B(\varphi_1) \wedge T2B(\varphi_2)$
- $T2B(\varphi_1 \vee \varphi_2) = T2B(\varphi_1) \vee T2B(\varphi_2)$
- $T2B(\neg\varphi_1) = \neg T2B(\varphi_1)$

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

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

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

$B2T(\varphi^P) = T2B^{-1}(\varphi^P)$

Offline DPLL(T)

```
Offline-DPLL_T(T-formula φ)
  φ^P ← T2B(φ)
  while (TRUE) do
    μ^P, res ← CDCL(φ^P)
    if res = UNSAT then return UNSAT
    else
      t, res ← T-solve(B2T(μ^P))
      if res = SAT then return SAT
      else φ^P ← φ^P ∧ T2B(t)
```

Offline DPLL(T)

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

T-solve can compute any clause t s.t.

1. φ ⇒ t
2. Atoms(t) ⊆ Atoms(φ)
3. T2B(t) conflicts with μ^P

Offline DPLL(T)

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

T-solve can compute any clause t s.t.

1. φ ⇒ t
2. Atoms(t) ⊆ Atoms(φ)
3. T2B(t) conflicts with μ^P

The requirements on t ensure soundness (1) and termination (2-3).

Offline DPLL(T)

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

T-solve can compute any clause t s.t.

1. φ ⇒ t
2. Atoms(t) ⊆ Atoms(φ)
3. T2B(t) conflicts with μ^P

The requirements on t ensure soundness (1) and termination (2-3).

What is a t that satisfies 1-3?

Offline DPLL(T)

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

T-solve can compute any clause t s.t.

1. φ ⇒ t
2. Atoms(t) ⊆ Atoms(φ)
3. T2B(t) conflicts with μ^P

The requirements on t ensure soundness (1) and termination (2-3).

What is a t that satisfies 1-3?

t = B2T(¬ μ^P)

Offline DPLL(T): Example

```
Offline-DPLL_T(T-formula φ)
  φ^P ← T2B(φ)
  while (TRUE) do
    μ^P, res ← CDCL(φ^P)
    if res = UNSAT then return UNSAT
    else
      t, res ← T-solve(B2T(μ^P))
      if res = SAT then return SAT
      else φ^P ← φ^P ∧ T2B(t)
```

Offline DPLL(T): Example

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

- φ ← (x = 1) ∧ ((x = 2) ∨ (x = 3))

Offline DPLL(T): Example

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

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

Offline DPLL(T): Example

Offline-DPLL$_T$(T-formula φ)
  $\varphi^P \leftarrow$ **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 res = SAT **then return** SAT
      else $\varphi^P \leftarrow \varphi^P \wedge$ **T2B**(t)

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

Offline DPLL(T): Example

Offline-DPLL$_T$(T-formula φ)
  $\varphi^P \leftarrow$ **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 res = SAT **then return** SAT
      else $\varphi^P \leftarrow \varphi^P \wedge$ **T2B**(t)

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

Offline DPLL(T): Example

Offline-DPLL$_T$(T-formula φ)
  $\varphi^P \leftarrow$ **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 res = SAT **then return** SAT
      else $\varphi^P \leftarrow \varphi^P \wedge$ **T2B**(t)

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

Offline DPLL(T): Example

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

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

Offline DPLL(T): Example

Offline-DPLL$_T$(T-formula φ)
  $\varphi^P \leftarrow$ **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 res = SAT **then return** SAT
      else $\varphi^P \leftarrow \varphi^P \wedge$ **T2B**(t)

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

Offline DPLL(T): Example

Offline-DPLL$_T$(T-formula φ)
  $\varphi^P \leftarrow$ **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 res = SAT **then return** SAT
      else $\varphi^P \leftarrow \varphi^P \wedge$ **T2B**(t)

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

Offline DPLL(T): Example

Offline-DPLL$_T$(T-formula φ)
  $\varphi^P \leftarrow$ **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 res = SAT **then return** SAT
      else $\varphi^P \leftarrow \varphi^P \wedge$ **T2B**(t)

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

Offline DPLL(T): Example

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

t = B2T(¬ μ^P) is too weak; it blocks one assignment at a time.

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

Offline DPLL(T): Example

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

t = B2T(¬ μ^P) is too weak; it blocks one assignment at a time.

What is a better t?

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

Offline DPLL(T): Example

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

t = B2T(¬ μ^P) is too weak; it blocks one assignment at a time.

What is a better t?

t = B2T(¬ MINIMALUNSATCORE(μ^P))

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

Offline DPLL(T): Example

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

t = B2T(¬ μ^P) is too weak; it blocks one assignment at a time.

What is a better t?

t = B2T(¬ MINIMALUNSATCORE(μ^P))

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

Offline DPLL(T): Example

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

t = B2T(¬ μ^P) is too weak; it blocks one assignment at a time.

What is a better t?

t = B2T(¬ MINIMALUNSATCORE(μ^P))

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

Online DPLL(T)

```
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
```

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
  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
```

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
  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
```

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
  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-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): Example

T_R-formula φ and φ^p:

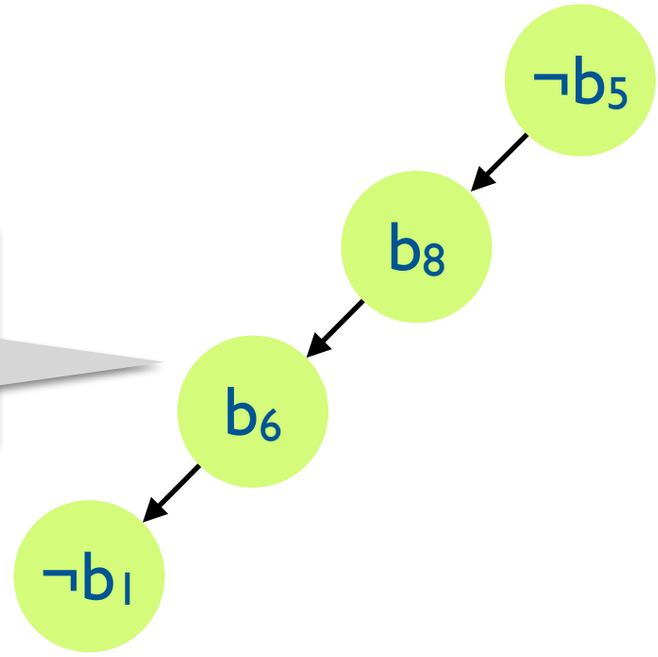
1. $\neg(2x_2 - x_3 > 2) \vee A_1$, $\neg b_1 \vee A_1$
2. $\neg A_2 \vee (x_1 - x_5 \leq 1)$, $\neg A_2 \vee b_2$
3. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $A_2 \vee b_3$
4. $\neg(2x_3 + x_4 \geq 5) \vee \neg(3x_1 - x_3 \leq 6) \vee \neg A_1$, $\neg b_4 \vee \neg b_5 \vee \neg A_1$
5. $(3x_1 - 2x_2 \leq 3)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

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. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $A_2 \vee b_3$
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. $(3x_1 - 2x_2 \leq 3) \vee A_1$, $A_1 \vee b_3$
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. $(x_3 = 3x_5 + 4) \vee A_1 \vee A_2$, $b_8 \vee A_2 \vee A_1$

T-DECIDE makes 4 decisions.



Online DPLL(T): T-DEDUCE

```
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
```

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 propagation*. 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
  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
```

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 propagation*. 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
  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
```

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 propagation*. 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
  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
```

Applies BCP to φ^P and μ^P until

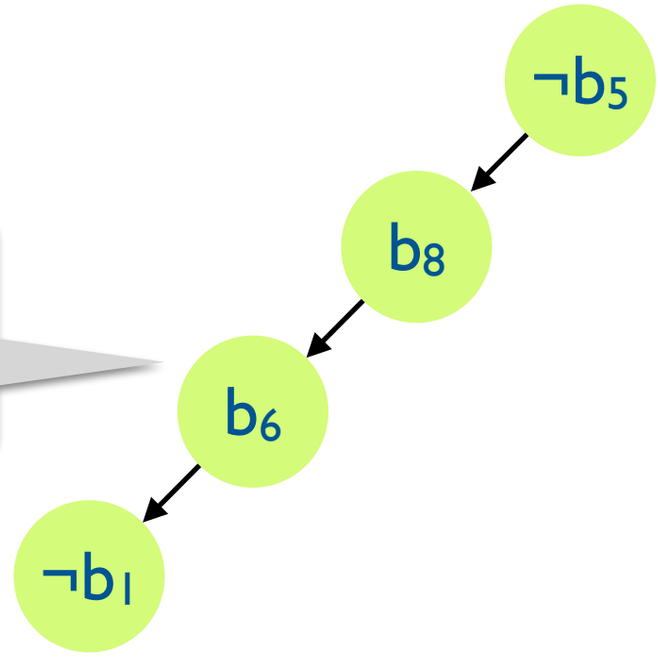
- μ^P propositionally violates φ^P: returns CONFLICT.
- μ^P propositionally satisfies φ^P: invokes T-solver on B2T(μ^P) and returns SAT if T-solver does. Otherwise returns CONFLICT.
- no more literals can be deduced: invokes T-solver on partial assignment B2T(μ^P) and returns CONFLICT is T-solver returns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

Online DPLL(T): Example

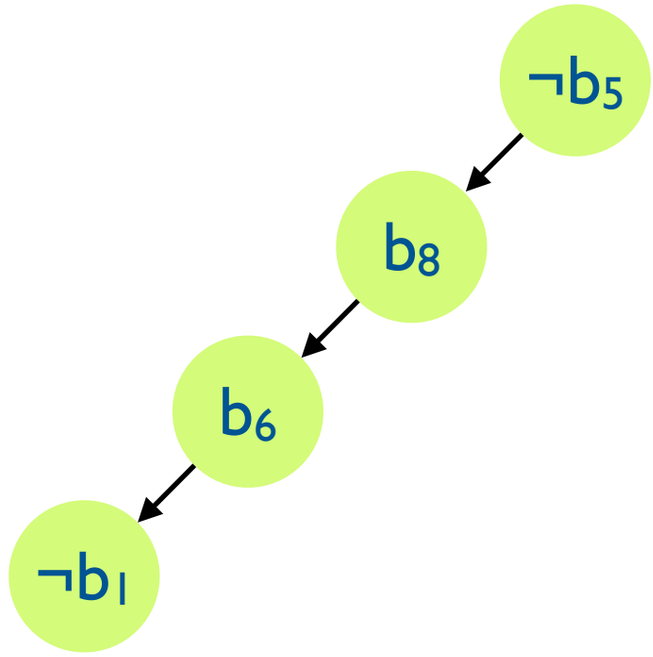
T_R-formula φ and φ^p:

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. $(3x_1 - 2x_2 \leq 3) \vee A_2, A_2 \vee b_3$
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. $(3x_1 - 2x_2 \leq 3) \vee A_1, A_1 \vee b_3$
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. $(x_3 = 3x_5 + 4) \vee A_1 \vee A_2, b_8 \vee A_2 \vee A_1$

T-DECIDE makes 4 decisions.



Online DPLL(T): Example



T_R-formula φ and φ^p:

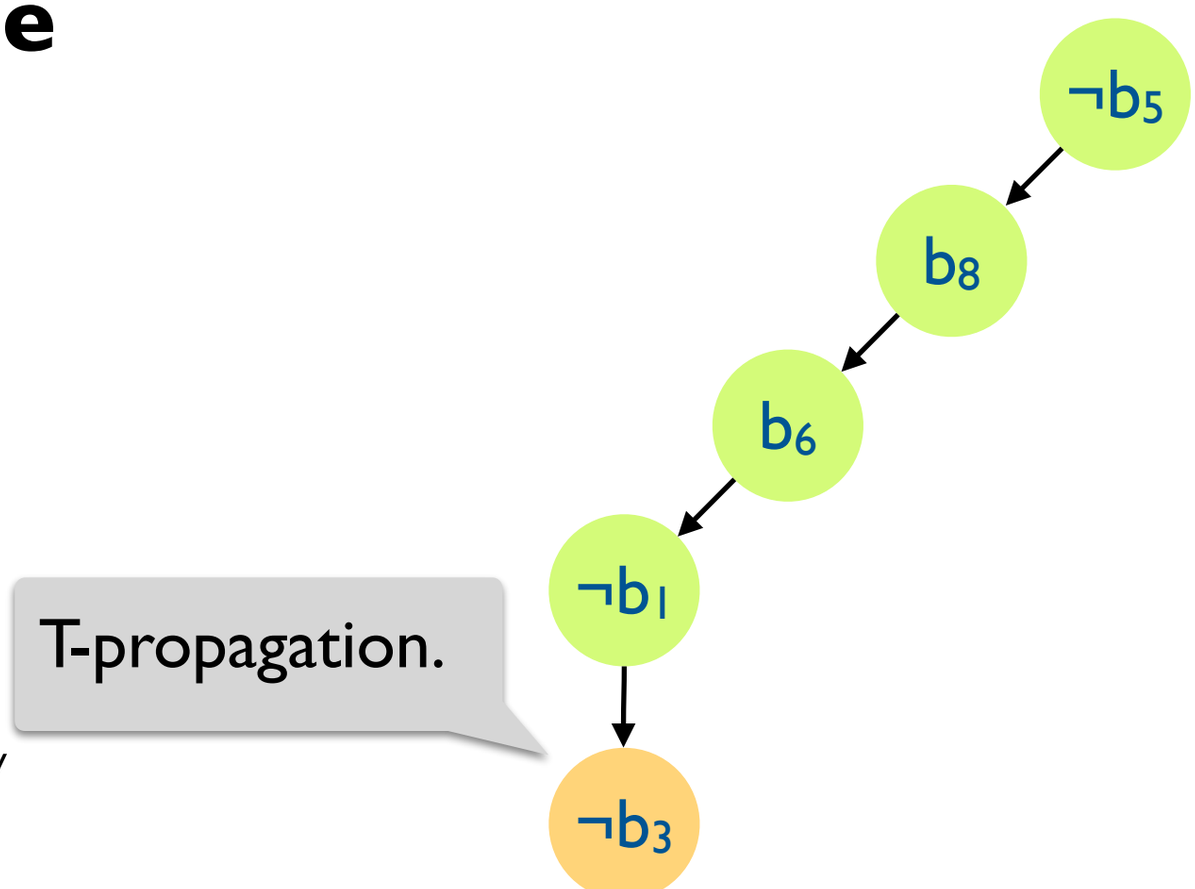
1. $\neg(2x_2 - x_3 > 2) \vee A_1$, $\neg b_1 \vee A_1$
2. $\neg A_2 \vee (x_1 - x_5 \leq 1)$, $\neg A_2 \vee b_2$
3. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $A_2 \vee b_3$
4. $\neg(2x_3 + x_4 \geq 5) \vee \neg(3x_1 - x_3 \leq 6) \vee \neg A_1$, $\neg b_4 \vee \neg b_5 \vee \neg A_1$
5. $(3x_1 - 2x_2 \leq 3)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

Early pruning.

Online DPLL(T): Example

T_R-formula φ and φ^p:

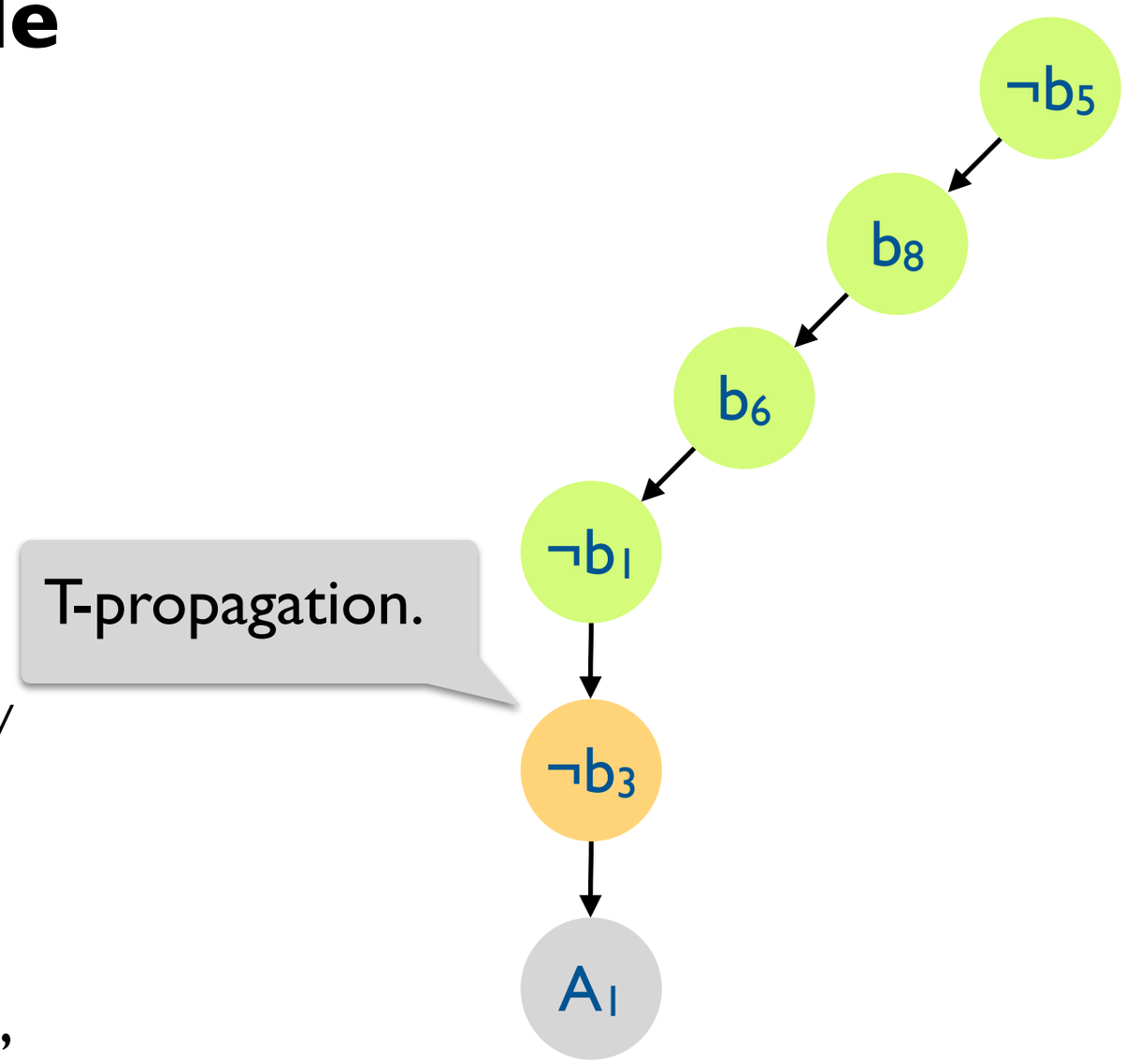
1. $\neg(2x_2 - x_3 > 2) \vee A_1$, $\neg b_1 \vee A_1$
2. $\neg A_2 \vee (x_1 - x_5 \leq 1)$, $\neg A_2 \vee b_2$
3. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $A_2 \vee b_3$
4. $\neg(2x_3 + x_4 \geq 5) \vee \neg(3x_1 - x_3 \leq 6) \vee \neg A_1$, $\neg b_4 \vee \neg b_5 \vee \neg A_1$
5. $(3x_1 - 2x_2 \leq 3)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



Online DPLL(T): Example

T_R-formula φ and φ^p:

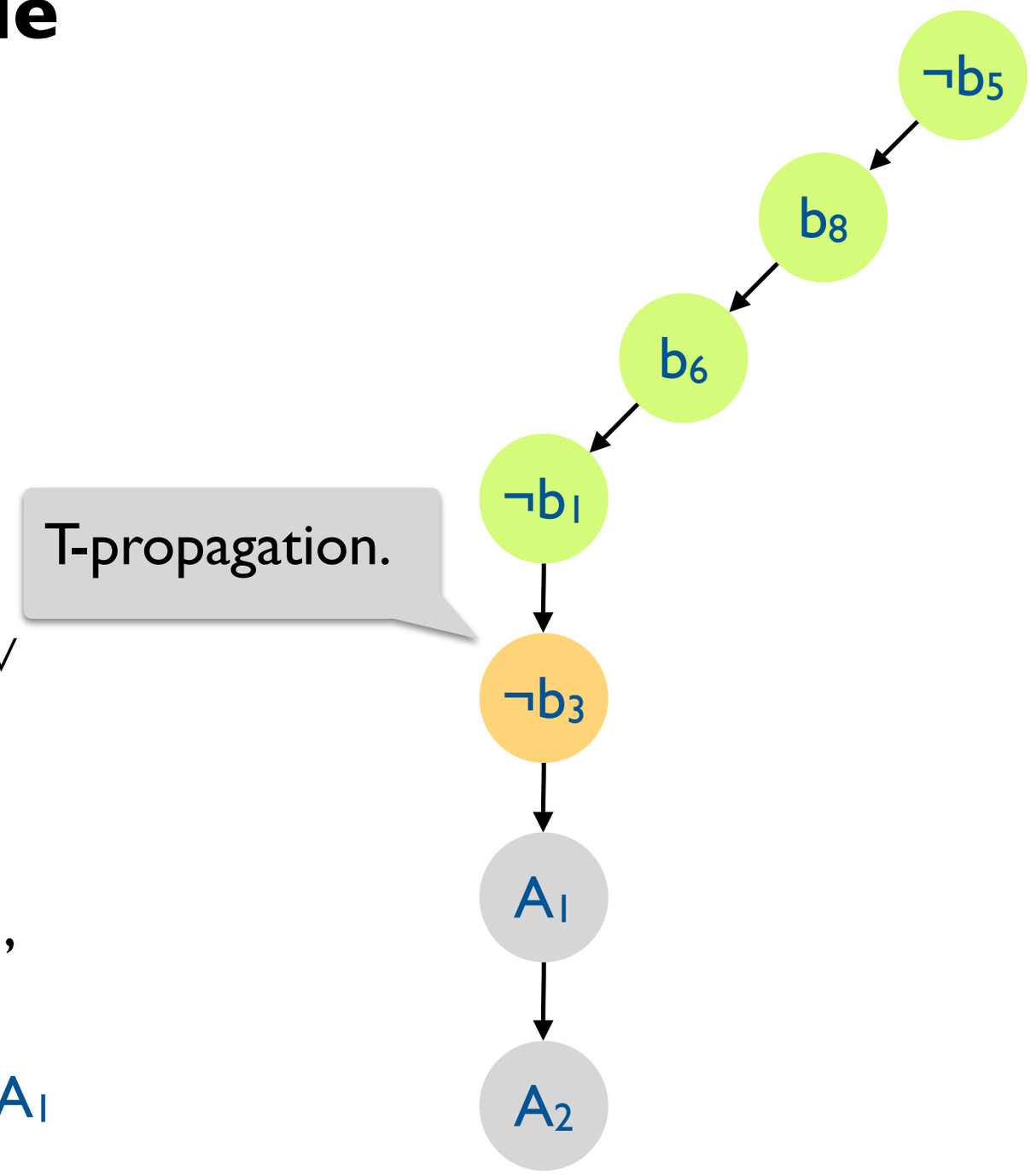
1. $\neg(2x_2 - x_3 > 2) \vee A_1$, $\neg b_1 \vee A_1$
2. $\neg A_2 \vee (x_1 - x_5 \leq 1)$, $\neg A_2 \vee b_2$
3. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $A_2 \vee b_3$
4. $\neg(2x_3 + x_4 \geq 5) \vee \neg(3x_1 - x_3 \leq 6) \vee \neg A_1$, $\neg b_4 \vee \neg b_5 \vee \neg A_1$
5. $(3x_1 - 2x_2 \leq 3)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



Online DPLL(T): Example

T_R-formula φ and φ^p:

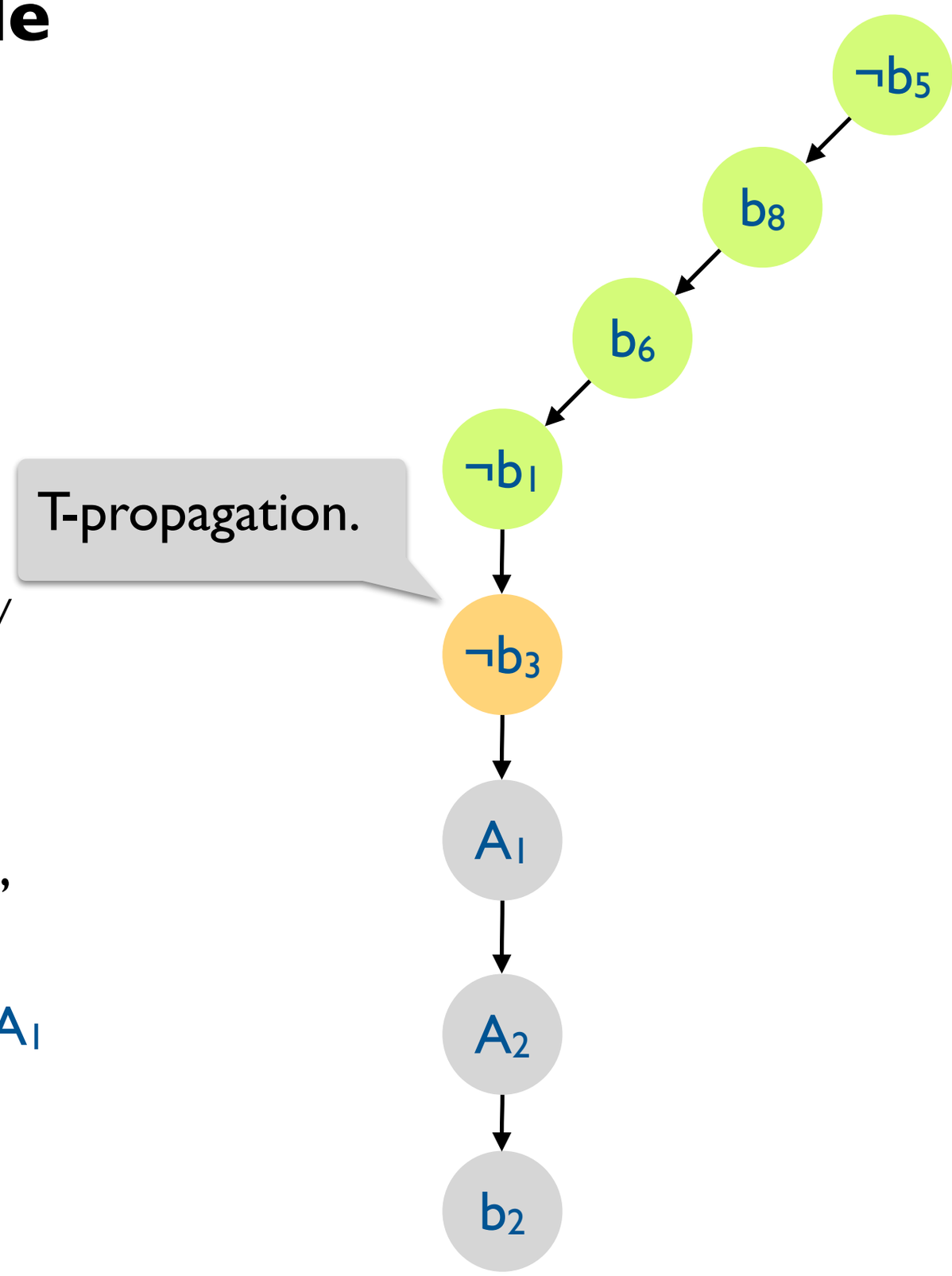
1. $\neg(2x_2 - x_3 > 2) \vee A_1$, $\neg b_1 \vee A_1$
2. $\neg A_2 \vee (x_1 - x_5 \leq 1)$, $\neg A_2 \vee b_2$
3. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $A_2 \vee b_3$
4. $\neg(2x_3 + x_4 \geq 5) \vee \neg(3x_1 - x_3 \leq 6) \vee \neg A_1$, $\neg b_4 \vee \neg b_5 \vee \neg A_1$
5. $(3x_1 - 2x_2 \leq 3)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



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. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $A_2 \vee b_3$
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. $(3x_1 - 2x_2 \leq 3) \vee A_1$, $A_1 \vee b_3$
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. $(x_3 = 3x_5 + 4) \vee A_1 \vee A_2$, $b_8 \vee A_2 \vee A_1$
8. $b_5 \vee b_1 \vee \neg b_3$



Online DPLL(T): T-AnalyzeConflict

```
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
```

Extends AnalyzeConflict from CDCL:

- if the conflict is caused by a boolean (BCP) failure, returns the same bevel and conflict clause AnalyzeConflict
- if the conflict is caused by a theory failure, returns a mixed boolean+theory conflict clause built from T2B(¬η), where η is the conflict set return by the T-solver.
- adds learned clauses to database (*T-learning*).

Online DPLL(T): T-AnalyzeConflict

```
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
```

Extends AnalyzeConflict from CDCL:

- if the conflict is caused by a boolean (BCP) failure, returns the same bevel and conflict clause AnalyzeConflict
- if the conflict is caused by a theory failure, returns a mixed boolean+theory conflict clause built from T2B(¬η), where η is the conflict set return by the T-solver.
- adds learned clauses to database (*T-learning*).

Online DPLL(T): T-AnalyzeConflict

```
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
```

Extends AnalyzeConflict from CDCL:

- if the conflict is caused by a boolean (BCP) failure, returns the same bevel and conflict clause AnalyzeConflict
- if the conflict is caused by a theory failure, returns a mixed boolean+theory conflict clause built from T2B(¬η), where η is the conflict set return by the T-solver.
- adds learned clauses to database (*T-learning*).

Online DPLL(T): T-AnalyzeConflict

```
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
```

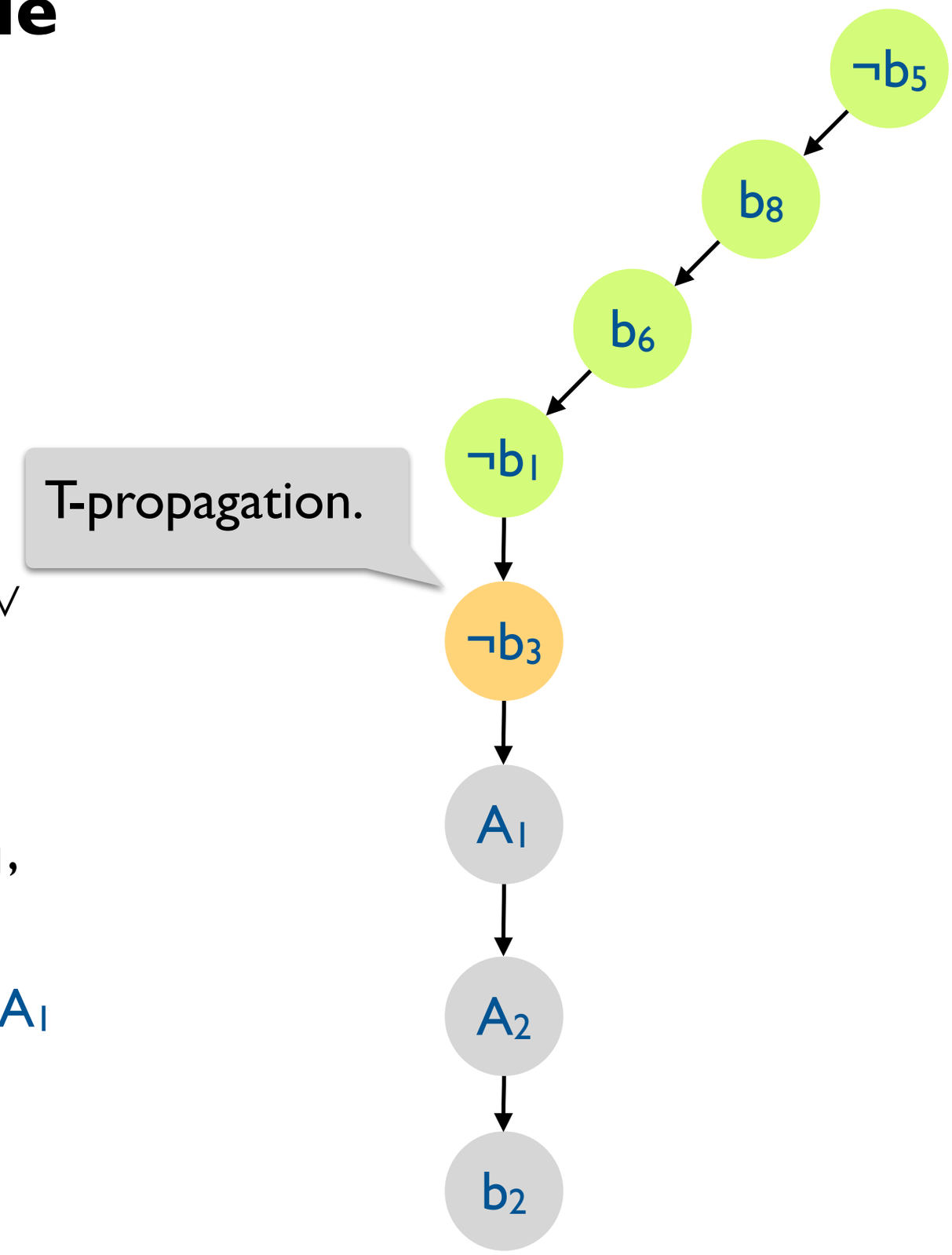
Extends AnalyzeConflict from CDCL:

- if the conflict is caused by a boolean (BCP) failure, returns the same bevel and conflict clause AnalyzeConflict
- if the conflict is caused by a theory failure, returns a mixed boolean+theory conflict clause built from T2B(¬η), where η is the conflict set return by the T-solver.
- adds learned clauses to database (*T-learning*).

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. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $A_2 \vee b_3$
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. $(3x_1 - 2x_2 \leq 3) \vee A_1$, $A_1 \vee b_3$
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. $(x_3 = 3x_5 + 4) \vee A_1 \vee A_2$, $b_8 \vee A_2 \vee A_1$
8. $b_5 \vee b_1 \vee \neg b_3$

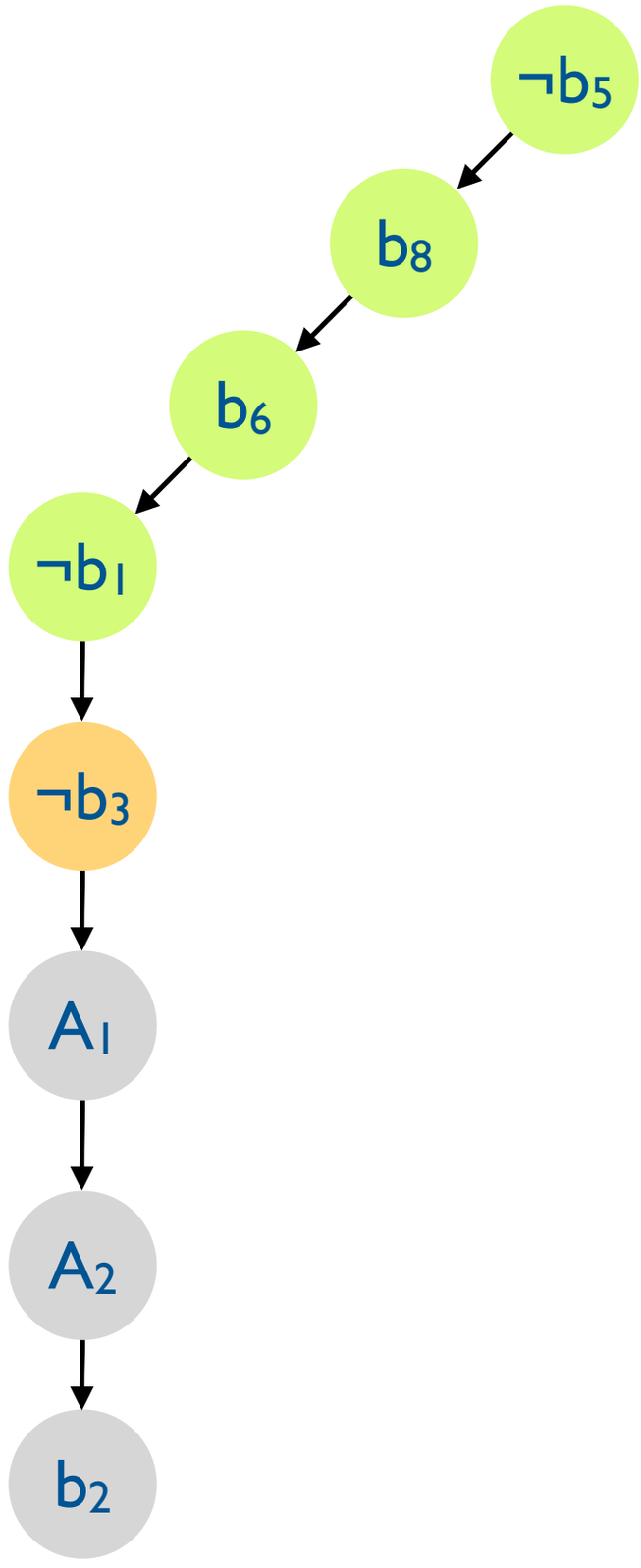


Online DPLL(T): Example

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. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $A_2 \vee b_3$
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. $(3x_1 - 2x_2 \leq 3) \vee A_1$, $A_1 \vee b_3$
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. $(x_3 = 3x_5 + 4) \vee A_1 \vee A_2$, $b_8 \vee A_2 \vee A_1$
8. $b_5 \vee b_1 \vee \neg b_3$
9. $b_5 \vee \neg b_8 \vee \neg b_2$

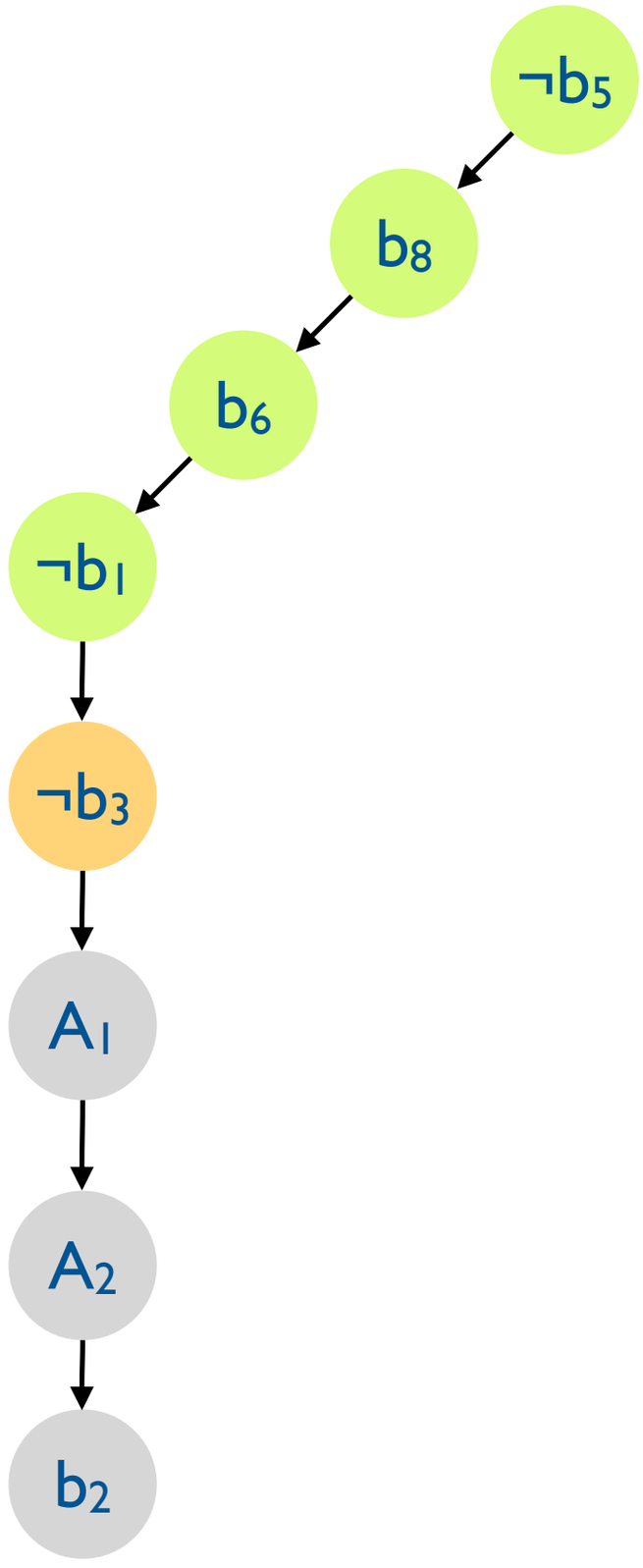
T conflict clause.



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. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $A_2 \vee b_3$
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. $(3x_1 - 2x_2 \leq 3) \vee A_1$, $A_1 \vee b_3$
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. $(x_3 = 3x_5 + 4) \vee A_1 \vee A_2$, $b_8 \vee A_2 \vee A_1$
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 clause.

Online DPLL(T): T-BACKTRACK

```
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 BACKTRACK in CDCL:

- 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
  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
```

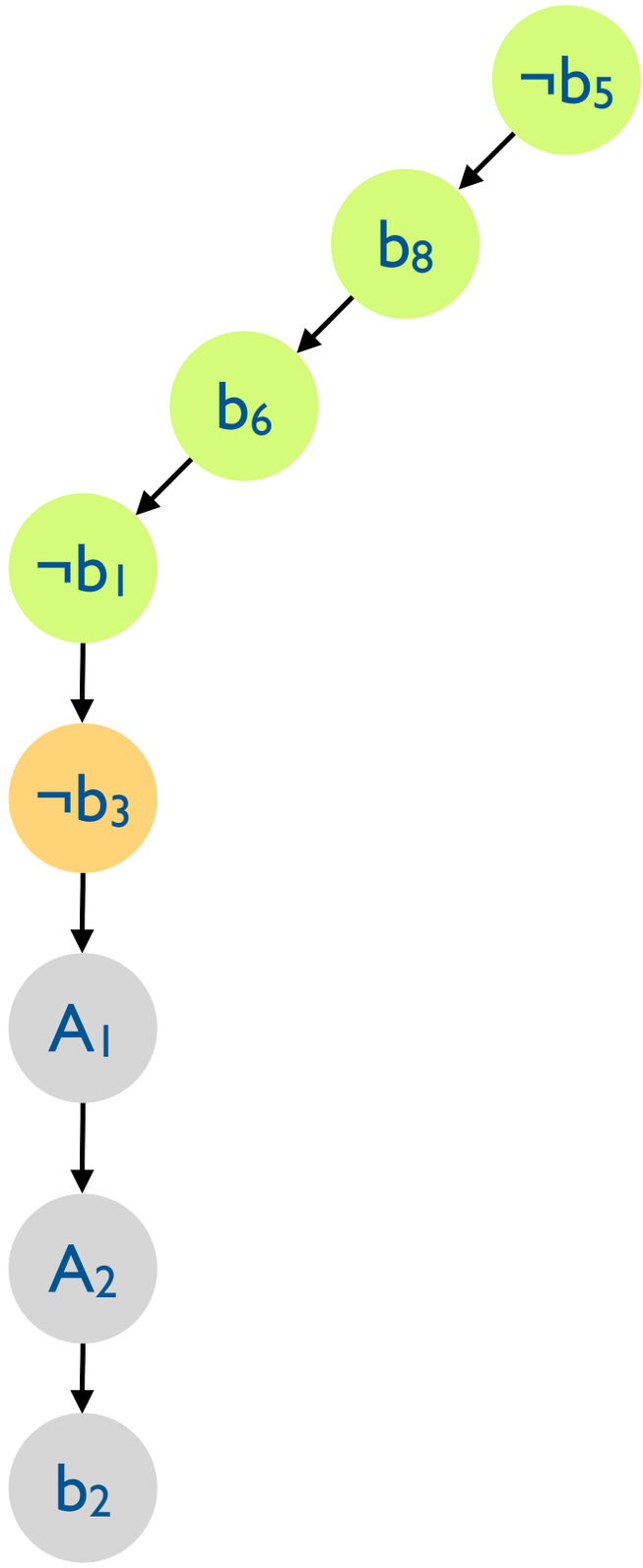
Analogous to BACKTRACK in CDCL:

- Backtracks to blevel by undoing all the assignments > blevel (*T-backjumping*).

Online DPLL(T): Example

T_R-formula φ and φ^p:

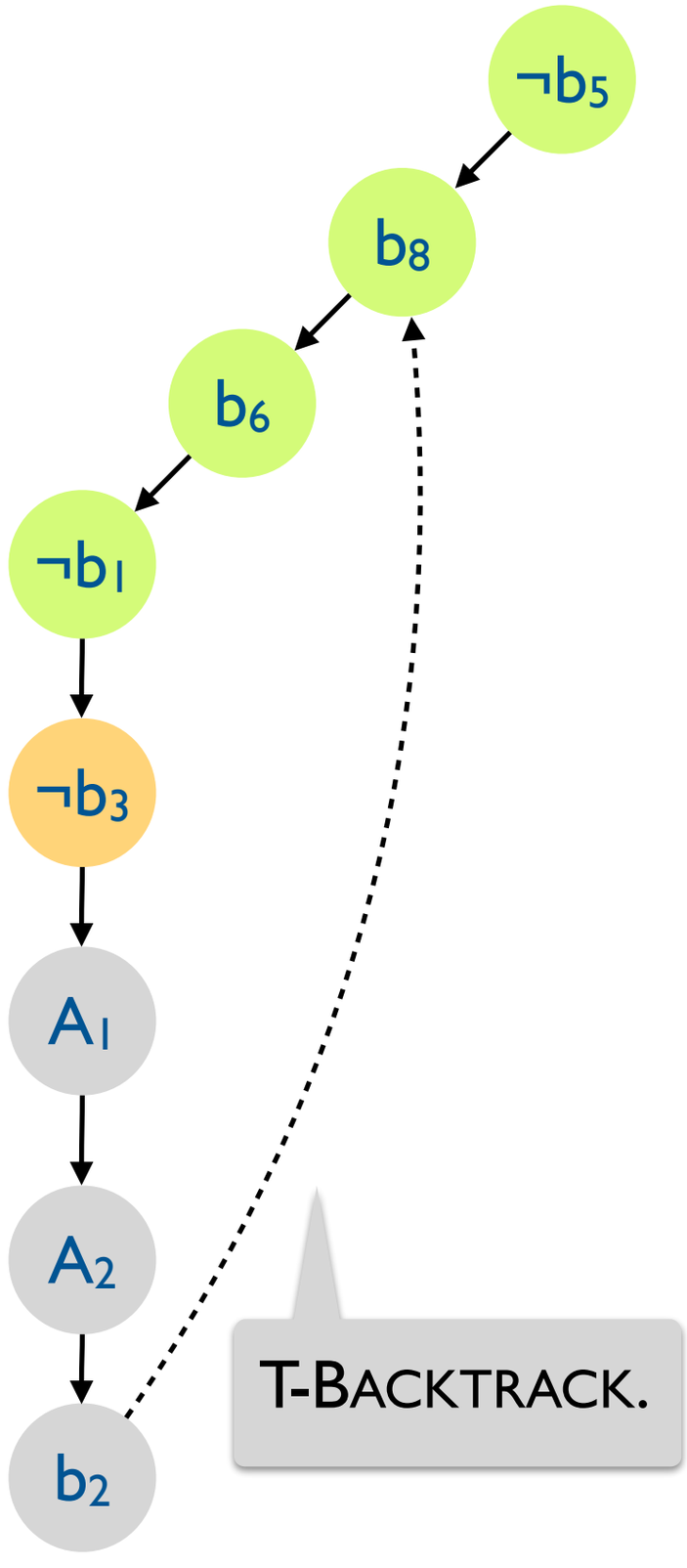
1. $\neg(2x_2 - x_3 > 2) \vee A_1$, $\neg b_1 \vee A_1$
2. $\neg A_2 \vee (x_1 - x_5 \leq 1)$, $\neg A_2 \vee b_2$
3. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $A_2 \vee b_3$
4. $\neg(2x_3 + x_4 \geq 5) \vee \neg(3x_1 - x_3 \leq 6) \vee \neg A_1$, $\neg b_4 \vee \neg b_5 \vee \neg A_1$
5. $(3x_1 - 2x_2 \leq 3)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



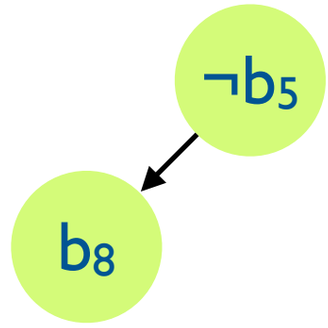
Online DPLL(T): Example

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. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $A_2 \vee b_3$
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. $(3x_1 - 2x_2 \leq 3) \vee A_1$, $A_1 \vee b_3$
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. $(x_3 = 3x_5 + 4) \vee A_1 \vee A_2$, $b_8 \vee A_2 \vee A_1$
8. $b_5 \vee b_1 \vee \neg b_3$
9. $b_5 \vee \neg b_8 \vee \neg b_2$
10. $b_5 \vee \neg b_8 \vee b_1$



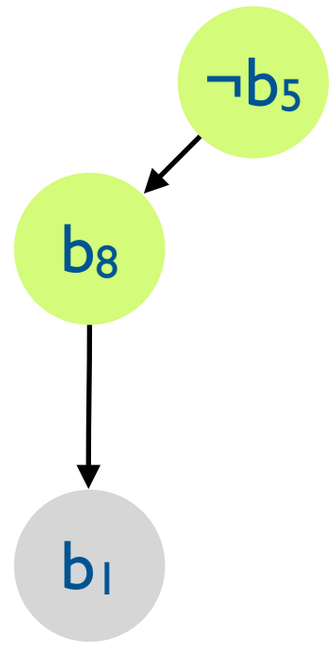
Online DPLL(T): Example



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. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $A_2 \vee b_3$
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. $(3x_1 - 2x_2 \leq 3) \vee A_1$, $A_1 \vee b_3$
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. $(x_3 = 3x_5 + 4) \vee A_1 \vee A_2$, $b_8 \vee A_2 \vee A_1$
8. $b_5 \vee b_1 \vee \neg b_3$
9. $b_5 \vee \neg b_8 \vee \neg b_2$
10. $b_5 \vee \neg b_8 \vee b_1$

Online DPLL(T): Example



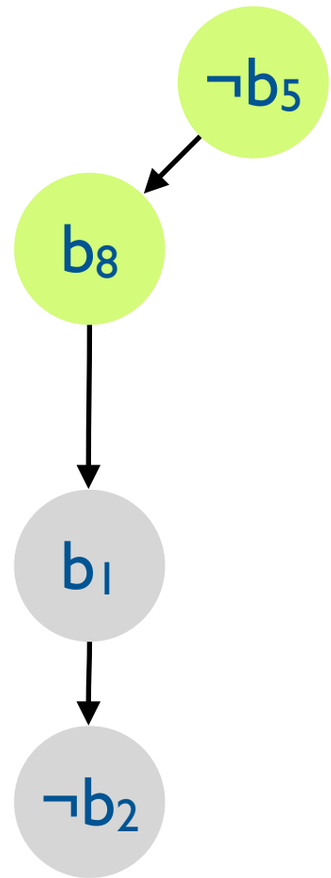
T_R-formula φ and φ^p:

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. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $A_2 \vee b_3$
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. $(3x_1 - 2x_2 \leq 3) \vee A_1$, $A_1 \vee b_3$
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. $(x_3 = 3x_5 + 4) \vee A_1 \vee A_2$, $b_8 \vee A_2 \vee A_1$
8. $b_5 \vee b_1 \vee \neg b_3$
9. $b_5 \vee \neg b_8 \vee \neg b_2$
10. $b_5 \vee \neg b_8 \vee b_1$

Online DPLL(T): Example

T_R-formula φ and φ^p:

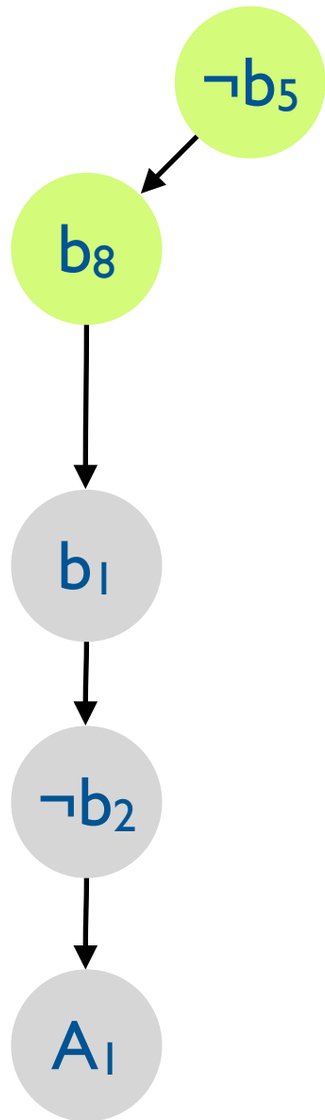
1. $\neg(2x_2 - x_3 > 2) \vee A_1$, $\neg b_1 \vee A_1$
2. $\neg A_2 \vee (x_1 - x_5 \leq 1)$, $\neg A_2 \vee b_2$
3. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $A_2 \vee b_3$
4. $\neg(2x_3 + x_4 \geq 5) \vee \neg(3x_1 - x_3 \leq 6) \vee \neg A_1$, $\neg b_4 \vee \neg b_5 \vee \neg A_1$
5. $(3x_1 - 2x_2 \leq 3)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



Online DPLL(T): Example

T_R-formula φ and φ^p:

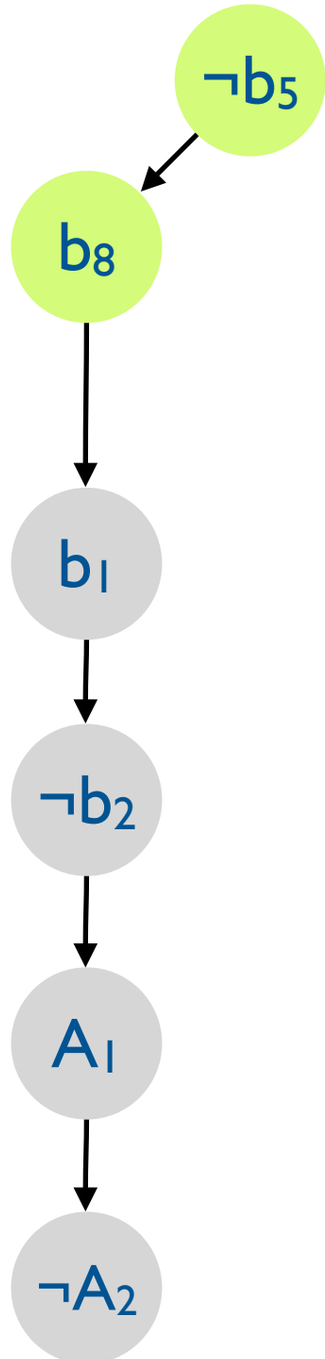
1. $\neg(2x_2 - x_3 > 2) \vee A_1$, $\neg b_1 \vee A_1$
2. $\neg A_2 \vee (x_1 - x_5 \leq 1)$, $\neg A_2 \vee b_2$
3. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $A_2 \vee b_3$
4. $\neg(2x_3 + x_4 \geq 5) \vee \neg(3x_1 - x_3 \leq 6) \vee \neg A_1$, $\neg b_4 \vee \neg b_5 \vee \neg A_1$
5. $(3x_1 - 2x_2 \leq 3)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



Online DPLL(T): Example

T_R-formula φ and φ^p:

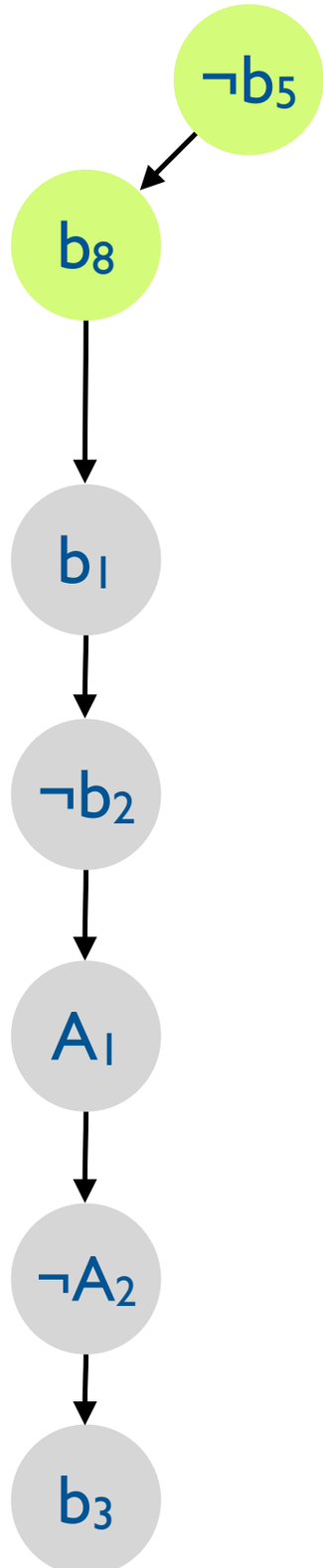
1. $\neg(2x_2 - x_3 > 2) \vee A_1$, $\neg b_1 \vee A_1$
2. $\neg A_2 \vee (x_1 - x_5 \leq 1)$, $\neg A_2 \vee b_2$
3. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $A_2 \vee b_3$
4. $\neg(2x_3 + x_4 \geq 5) \vee \neg(3x_1 - x_3 \leq 6) \vee \neg A_1$, $\neg b_4 \vee \neg b_5 \vee \neg A_1$
5. $(3x_1 - 2x_2 \leq 3)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



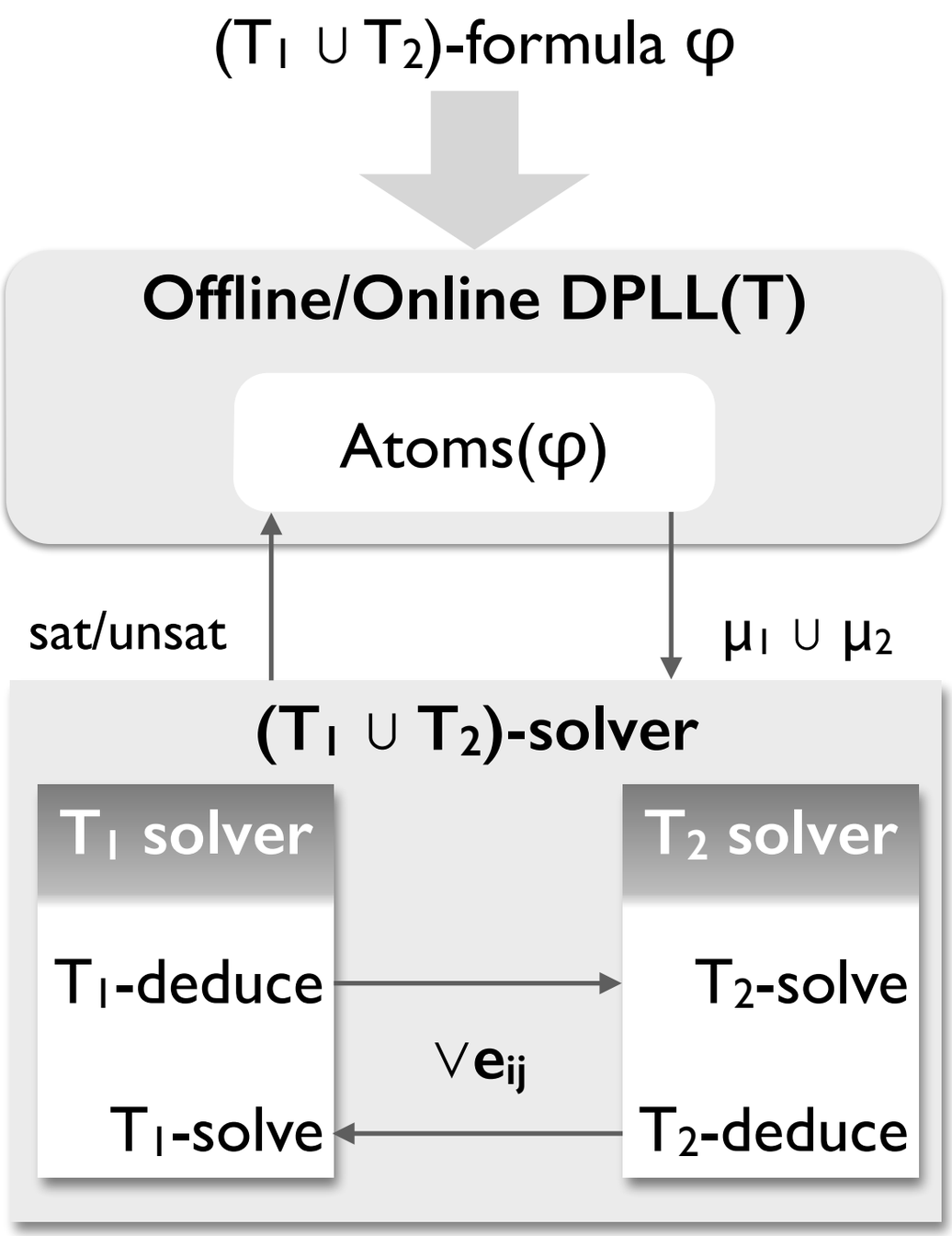
Online DPLL(T): Example

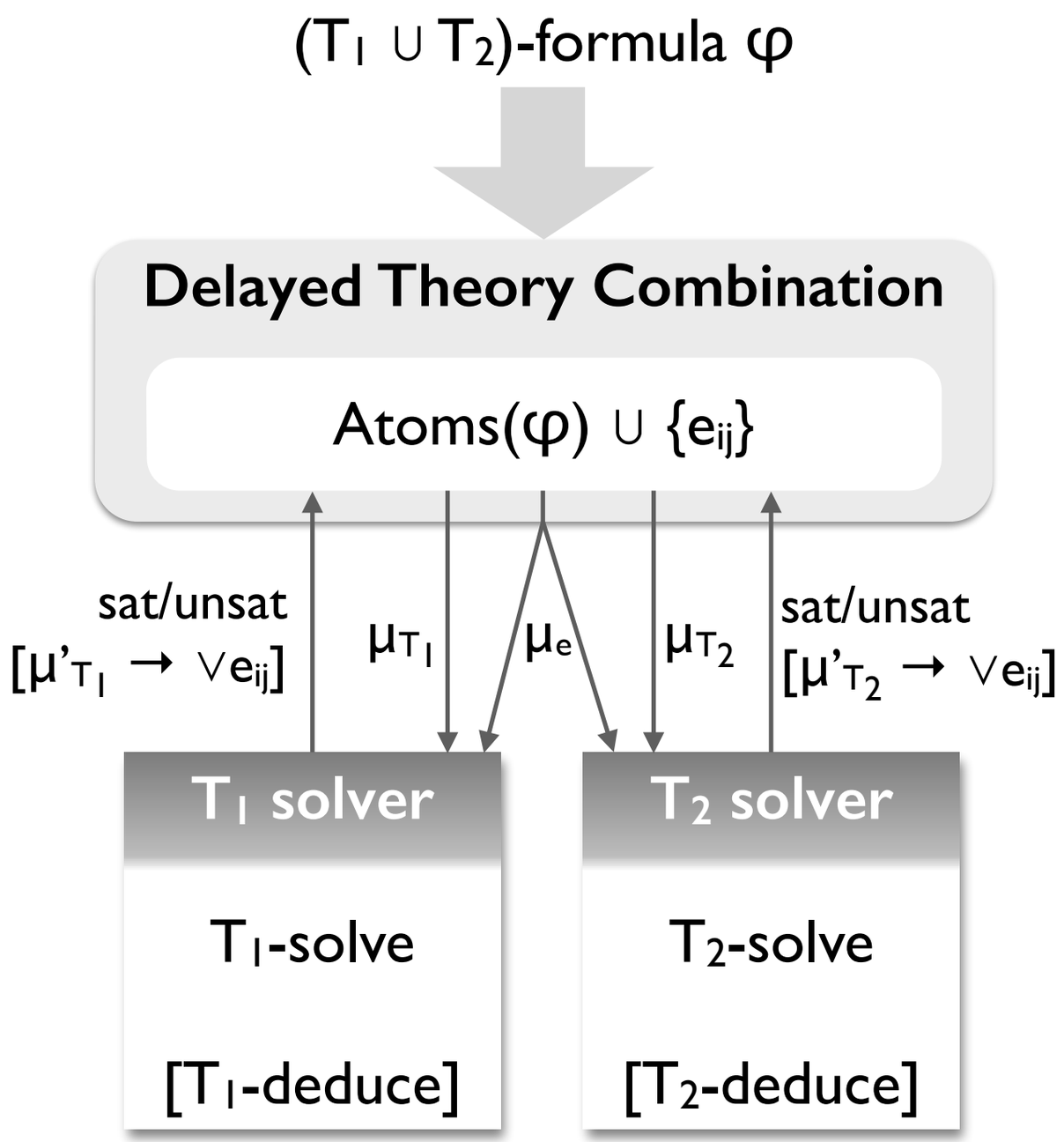
T_R-formula φ and φ^p:

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. $(3x_1 - 2x_2 \leq 3) \vee A_2$, $A_2 \vee b_3$
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. $(3x_1 - 2x_2 \leq 3) \vee A_1$, $A_1 \vee b_3$
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. $(x_3 = 3x_5 + 4) \vee A_1 \vee A_2$, $b_8 \vee A_2 \vee A_1$
8. $b_5 \vee b_1 \vee \neg b_3$
9. $b_5 \vee \neg b_8 \vee \neg b_2$
10. $b_5 \vee \neg b_8 \vee b_1$



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
```

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

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.

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.

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.

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 set (possibly containing interface equalities) from one of the T_i-solvers.

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 set (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!