

Computer-Aided Reasoning for Software

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

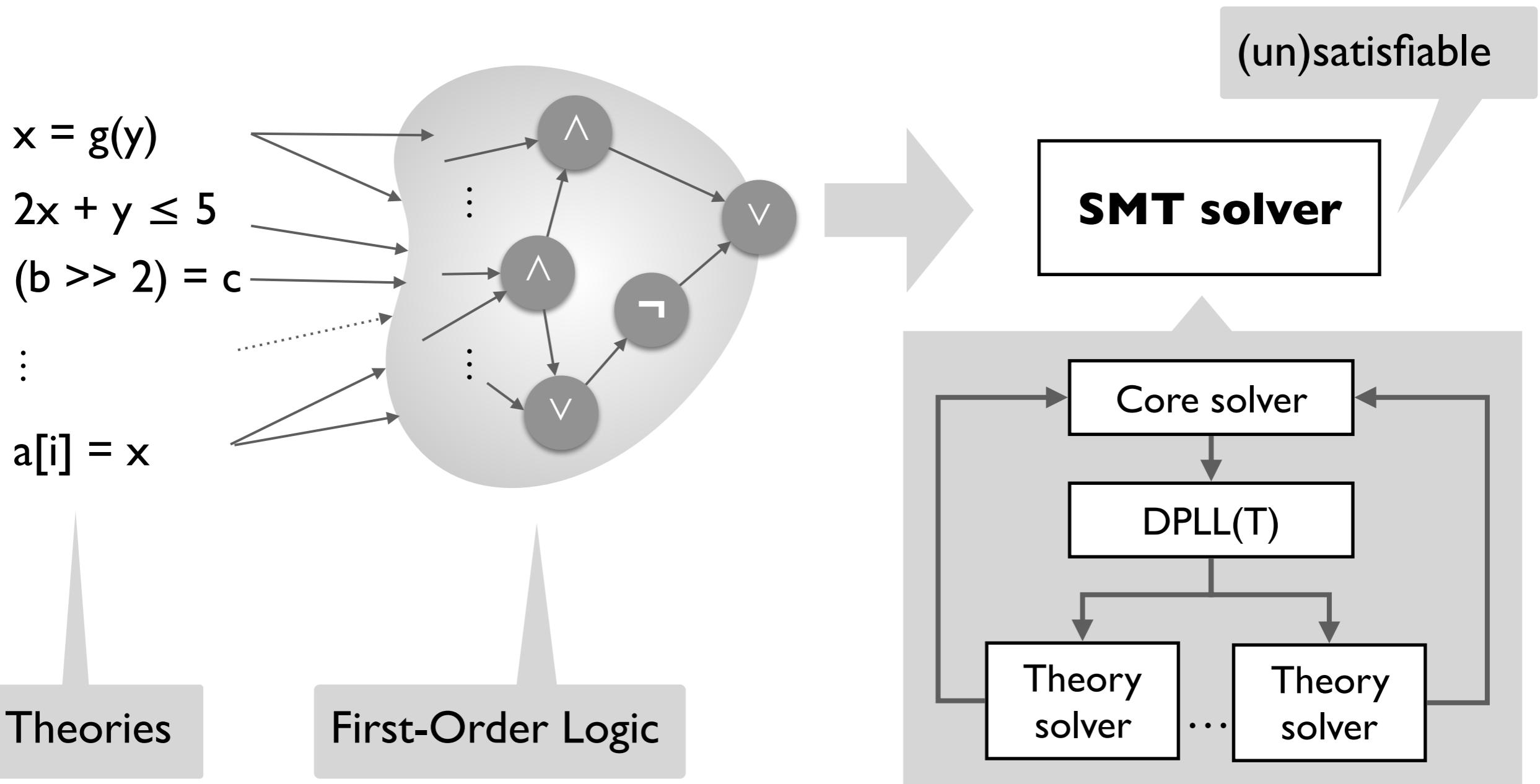
- Deciding arbitrary boolean combinations of theory constraints

Reminders

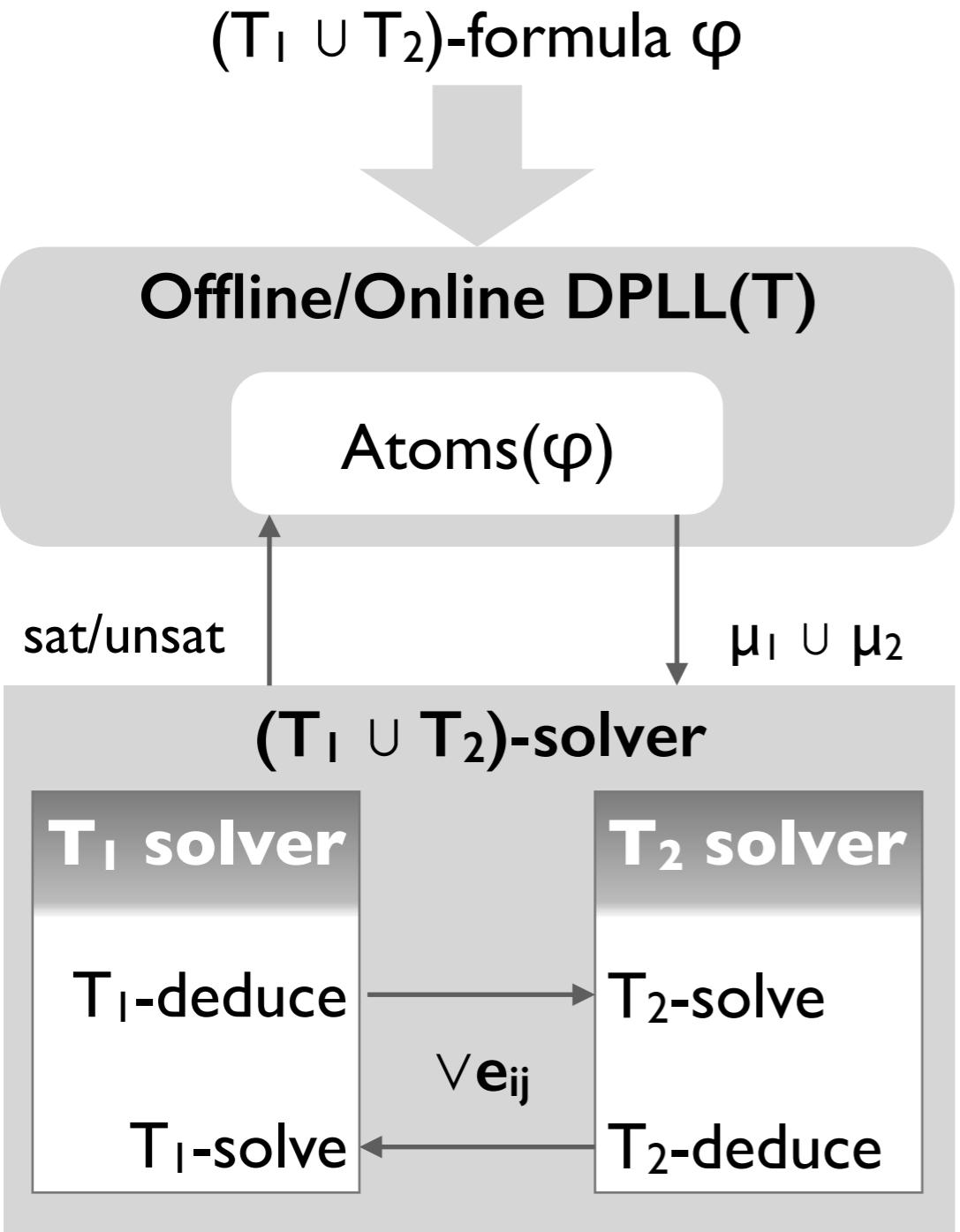
- Project proposal due on Thursday.



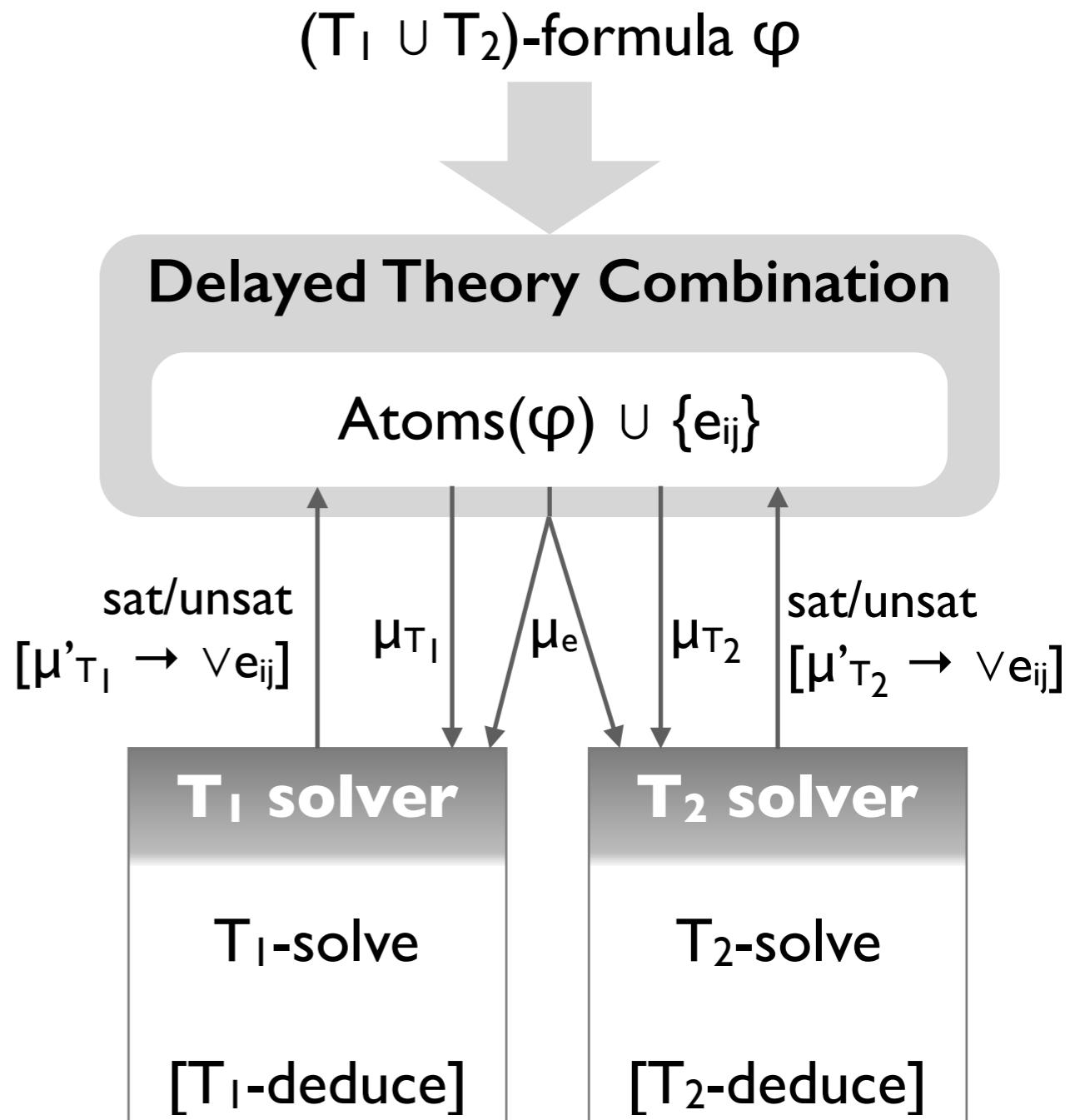
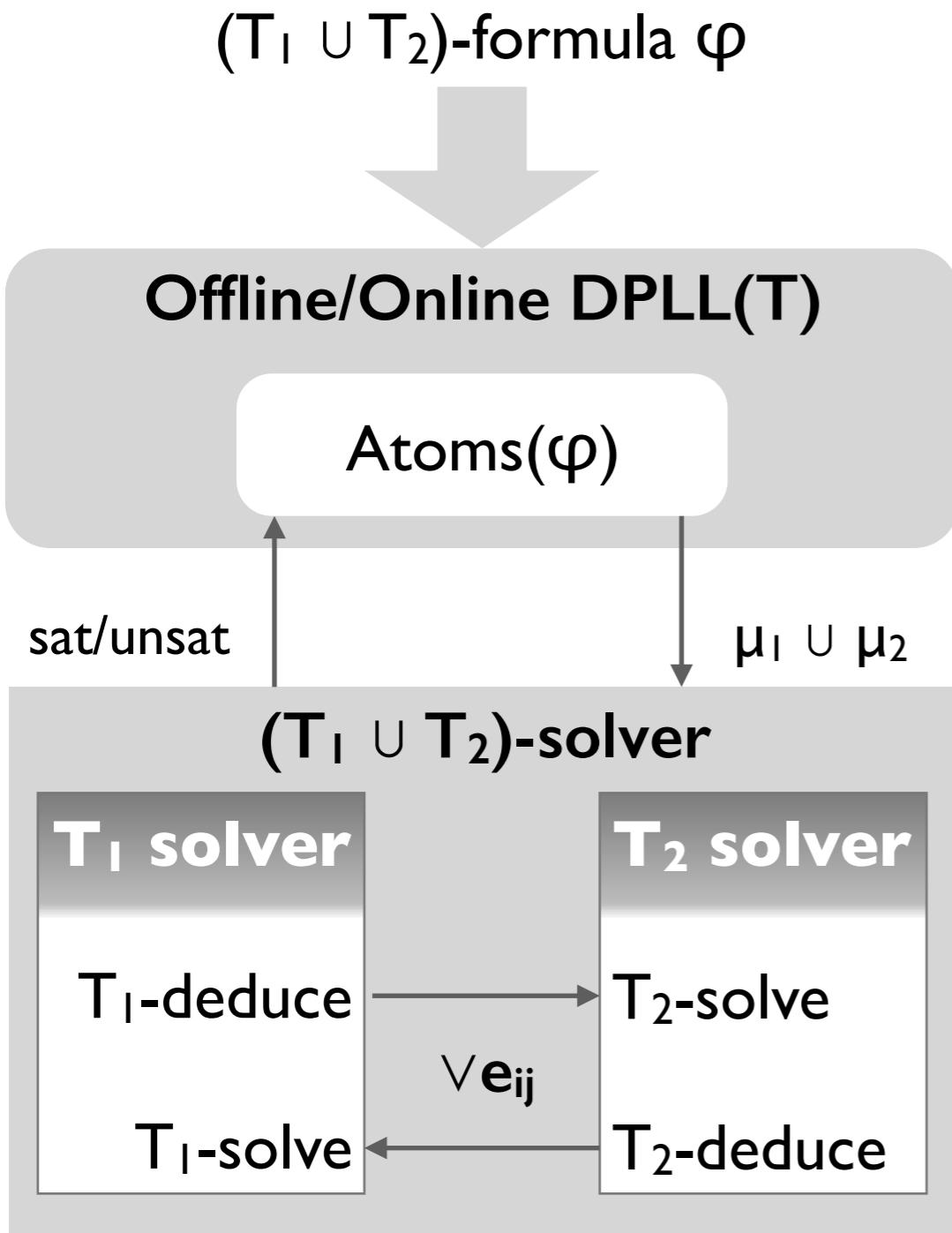
Recall: Satisfiability Modulo Theories (SMT)



The DPLL(T) Framework



The DPLL(T) Framework



Offline DPLL(T)

Offline-DPLL_T(T-formula φ)

```
 $\varphi^P \leftarrow \text{T2B}(\varphi)$ 
while (TRUE) do
     $\mu^P, \text{res} \leftarrow \text{CDCL}(\varphi^P)$ 
    if res = UNSAT then return UNSAT
    else
        T-res  $\leftarrow$  T-solve(B2T( $\mu^P$ ))
        if T-res = SAT then return SAT
        else  $\varphi^P \leftarrow \varphi^P \wedge \neg\mu^P$ 
```

Offline DPLL(T)

Assume φ is in CNF.

Offline-DPLL_T(T-formula φ)

```
 $\varphi^P \leftarrow \text{T2B}(\varphi)$ 
while (TRUE) do
     $\mu^P, \text{res} \leftarrow \text{CDCL}(\varphi^P)$ 
    if res = UNSAT then return UNSAT
    else
        T-res  $\leftarrow$  T-solve(B2T( $\mu^P$ ))
        if T-res = SAT then return SAT
        else  $\varphi^P \leftarrow \varphi^P \wedge \neg\mu^P$ 
```

Offline DPLL(T)

Assume φ is in CNF.

Offline-DPLL_T(T-formula φ)

```
 $\varphi^P \leftarrow \text{T2B}(\varphi)$ 
while (TRUE) do
     $\mu^P, \text{res} \leftarrow \text{CDCL}(\varphi^P)$ 
    if res = UNSAT then return UNSAT
    else
        T-res  $\leftarrow$  T-solve(B2T( $\mu^P$ ))
        if T-res = SAT then return SAT
        else  $\varphi^P \leftarrow \varphi^P \wedge \neg \mu^P$ 
```

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

Offline DPLL(T)

Assume φ is in CNF.

Offline-DPLL_T(T-formula φ)

```
 $\varphi^P \leftarrow \text{T2B}(\varphi)$ 
while (TRUE) do
     $\mu^P, \text{res} \leftarrow \text{CDCL}(\varphi^P)$ 
    if res = UNSAT then return UNSAT
    else
        T-res  $\leftarrow$  T-solve(B2T( $\mu^P$ ))
        if T-res = SAT then return SAT
        else  $\varphi^P \leftarrow \varphi^P \wedge \neg \mu^P$ 
```

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-DPLL $_T$ (T-formula φ)

```
 $\varphi^P \leftarrow \text{T2B}(\varphi)$ 
while (TRUE) do
     $\mu^P, \text{res} \leftarrow \text{CDCL}(\varphi^P)$ 
    if res = UNSAT then return UNSAT
    else
        T-res  $\leftarrow$  T-solve(B2T( $\mu^P$ ))
        if T-res = SAT then return SAT
        else  $\varphi^P \leftarrow \varphi^P \wedge \neg \mu^P$ 
```

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 .

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
- $T2B(\varphi_1 \wedge \varphi_2) = T2B(\varphi_1) \wedge T2B(\varphi_2)$
- $T2B(\varphi_1 \vee \varphi_2) = T2B(\varphi_1) \vee T2B(\varphi_2)$
- $T2B(\neg\varphi_1) = \neg T2B(\varphi_1)$

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

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

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

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

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

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

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

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

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

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

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

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

B2T(φ^p) = T2B⁻¹(φ^p)

Soundness and termination of offline DPLL(\mathbf{T})

Offline-DPLL $_{\mathbf{T}}$ (T-formula φ)

```
 $\varphi^P \leftarrow \text{T2B}(\varphi)$ 
while (TRUE) do
   $\mu^P, \text{res} \leftarrow \text{CDCL}(\varphi^P)$ 
  if res = UNSAT then return UNSAT
  else
    T-res  $\leftarrow$  T-solve(B2T( $\mu^P$ ))
    if T-res = SAT then return SAT
    else  $\varphi^P \leftarrow \varphi^P \wedge \neg \mu^P$ 
```

Soundness and termination of offline DPLL(\mathbf{T})

Offline-DPLL $_{\mathbf{T}}$ (T-formula φ)

```
 $\varphi^P \leftarrow \text{T2B}(\varphi)$ 
while (TRUE) do
     $\mu^P, \text{res} \leftarrow \text{CDCL}(\varphi^P)$ 
    if res = UNSAT then return UNSAT
    else
        T-res  $\leftarrow$  T-solve(B2T( $\mu^P$ ))
        if T-res = SAT then return SAT
        else  $\varphi^P \leftarrow \varphi^P \wedge \neg \mu^P$ 
```

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(\mathbf{T})

Offline-DPLL $_{\mathbf{T}}$ (T-formula φ)

```
 $\varphi^P \leftarrow \text{T2B}(\varphi)$ 
while (TRUE) do
     $\mu^P, \text{res} \leftarrow \text{CDCL}(\varphi^P)$ 
    if res = UNSAT then return UNSAT
    else
        T-res  $\leftarrow$  T-solve(B2T( $\mu^P$ ))
        if T-res = SAT then return SAT
        else  $\varphi^P \leftarrow \varphi^P \wedge \neg \mu^P$ 
```

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.

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)

```
 $\varphi^P \leftarrow \text{T2B}(\varphi)$ 
while (TRUE) do
   $\mu^P, \text{res} \leftarrow \text{CDCL}(\varphi^P)$ 
  if res = UNSAT then return UNSAT
  else
    T-res  $\leftarrow$  T-solve(B2T( $\mu^P$ ))
    if T-res = SAT then return SAT
    else  $\varphi^P \leftarrow \varphi^P \wedge \neg\mu^P$ 
```

Offline DPLL(T) example

Offline-DPLL_T(T-formula φ)

```
 $\varphi^P \leftarrow \text{T2B}(\varphi)$ 
while (TRUE) do
   $\mu^P, \text{res} \leftarrow \text{CDCL}(\varphi^P)$ 
  if res = UNSAT then return UNSAT
  else
    T-res  $\leftarrow$  T-solve(B2T( $\mu^P$ ))
    if T-res = SAT then return SAT
    else  $\varphi^P \leftarrow \varphi^P \wedge \neg\mu^P$ 
```

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

Offline DPLL(T) example

Offline-DPLL_T(T-formula φ)

$\varphi^P \leftarrow \text{T2B}(\varphi)$

while (TRUE) **do**

$\mu^P, \text{res} \leftarrow \text{CDCL}(\varphi^P)$

if res = UNSAT **then return** UNSAT

else

$T\text{-res} \leftarrow T\text{-solve}(\text{B2T}(\mu^P))$

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

else $\varphi^P \leftarrow \varphi^P \wedge \neg \mu^P$

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

Offline DPLL(T) example

Offline-DPLL_T(T-formula φ)

```
 $\varphi^P \leftarrow \text{T2B}(\varphi)$ 
while (TRUE) do
   $\mu^P, \text{res} \leftarrow \text{CDCL}(\varphi^P)$ 
  if res = UNSAT then return UNSAT
  else
    T-res  $\leftarrow$  T-solve(B2T( $\mu^P$ ))
    if T-res = SAT then return SAT
    else  $\varphi^P \leftarrow \varphi^P \wedge \neg \mu^P$ 
```

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

Offline DPLL(T) example

Offline-DPLL_T(T-formula φ)

$\varphi^P \leftarrow \text{T2B}(\varphi)$

while (TRUE) **do**

$\mu^P, \text{res} \leftarrow \text{CDCL}(\varphi^P)$

if res = UNSAT **then return** UNSAT

else

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

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

else $\varphi^P \leftarrow \varphi^P \wedge \neg \mu^P$

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

Offline DPLL(T) example

Offline-DPLL_T(T-formula φ)

$\varphi^P \leftarrow \text{T2B}(\varphi)$

while (TRUE) **do**

$\mu^P, \text{res} \leftarrow \text{CDCL}(\varphi^P)$

if res = UNSAT **then return** UNSAT

else

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

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

else $\varphi^P \leftarrow \varphi^P \wedge \neg\mu^P$

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

Offline DPLL(T) example

Offline-DPLL_T(T-formula φ)

$\varphi^P \leftarrow \text{T2B}(\varphi)$

while (TRUE) **do**

$\mu^P, \text{res} \leftarrow \text{CDCL}(\varphi^P)$

if res = UNSAT **then return** UNSAT

else

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

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

else $\varphi^P \leftarrow \varphi^P \wedge \neg\mu^P$

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

Offline DPLL(T) example

Offline-DPLL_T(T-formula φ)

$\varphi^P \leftarrow \text{T2B}(\varphi)$

while (TRUE) **do**

$\mu^P, \text{res} \leftarrow \text{CDCL}(\varphi^P)$

if res = UNSAT **then return** UNSAT

else

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

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

else $\varphi^P \leftarrow \varphi^P \wedge \neg\mu^P$

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

Offline DPLL(T) example

Offline-DPLL_T(T-formula φ)

$\varphi^P \leftarrow \text{T2B}(\varphi)$

while (TRUE) **do**

$\mu^P, \text{res} \leftarrow \text{CDCL}(\varphi^P)$

if res = UNSAT **then return** UNSAT

else

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

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

else $\varphi^P \leftarrow \varphi^P \wedge \neg\mu^P$

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

Offline DPLL(T) example

Offline-DPLL_T(T-formula φ)

$\varphi^P \leftarrow \text{T2B}(\varphi)$

while (TRUE) **do**

$\mu^P, \text{res} \leftarrow \text{CDCL}(\varphi^P)$

if res = UNSAT **then return** UNSAT

else

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

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

else $\varphi^P \leftarrow \varphi^P \wedge \neg\mu^P$

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

Offline DPLL(T): improvements

Offline-DPLL_T(T-formula φ)

$\varphi^P \leftarrow \text{T2B}(\varphi)$

while (TRUE) **do**

$\mu^P, \text{res} \leftarrow \text{CDCL}(\varphi^P)$

if res = UNSAT **then return** UNSAT

else

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

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

else $\varphi^P \leftarrow \varphi^P \wedge \neg\mu^P$

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

Offline DPLL(T): improvements

Offline-DPLL_T(T-formula φ)

```
 $\varphi^P \leftarrow \text{T2B}(\varphi)$ 
while (TRUE) do
   $\mu^P, \text{res} \leftarrow \text{CDCL}(\varphi^P)$ 
  if res = UNSAT then return UNSAT
  else
    T-res  $\leftarrow$  T-solve(B2T( $\mu^P$ ))
    if T-res = SAT then return SAT
    else  $\varphi^P \leftarrow \varphi^P \wedge \neg\mu^P$ 
```

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

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

Offline DPLL(T): improvements

Offline-DPLL_T(T-formula φ)

```
 $\varphi^P \leftarrow \text{T2B}(\varphi)$ 
while (TRUE) do
   $\mu^P, \text{res} \leftarrow \text{CDCL}(\varphi^P)$ 
  if res = UNSAT then return UNSAT
  else
    T-res  $\leftarrow$  T-solve(B2T( $\mu^P$ ))
    if T-res = SAT then return SAT
    else  $\varphi^P \leftarrow \varphi^P \wedge \neg\mu^P$ 
```

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

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

$\neg\text{T2B}(\text{MINIMALUNSATCORE}(\text{B2T}(\mu^P)))$

Offline DPLL(T): improvements

Offline-DPLL_T(T-formula φ)

```
 $\varphi^P \leftarrow \text{T2B}(\varphi)$ 
while (TRUE) do
   $\mu^P, \text{res} \leftarrow \text{CDCL}(\varphi^P)$ 
  if res = UNSAT then return UNSAT
  else
    T-res  $\leftarrow$  T-solve(B2T( $\mu^P$ ))
    if T-res = SAT then return SAT
    else
      t  $\leftarrow$  T2B(MINUNSATCORE(B2T( $\mu^P$ )))
       $\varphi^P \leftarrow \varphi^P \wedge \neg t$ 
```

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

Offline DPLL(T): improvements

Offline-DPLL_T(T-formula φ)

```
 $\varphi^P \leftarrow \text{T2B}(\varphi)$ 
while (TRUE) do
   $\mu^P, \text{res} \leftarrow \text{CDCL}(\varphi^P)$ 
  if res = UNSAT then return UNSAT
  else
    T-res  $\leftarrow$  T-solve(B2T( $\mu^P$ ))
    if T-res = SAT then return SAT
    else
      t  $\leftarrow$  T2B(MINUNSATCORE(B2T( $\mu^P$ )))
       $\varphi^P \leftarrow \varphi^P \wedge \neg t$ 
```

- $\varphi \leftarrow (x = 1) \wedge ((x = 2) \vee (x = 3))$
- $\varphi^P \leftarrow b_1 \wedge (b_2 \vee b_3)$
- $\mu^P \leftarrow b_1 \wedge b_2 \wedge b_3$
 - $\varphi^P \leftarrow b_1 \wedge (b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_2)$
 - $\mu^P \leftarrow b_1 \wedge \neg b_2 \wedge b_3$
 - $\varphi^P \leftarrow b_1 \wedge (b_2 \vee b_3) \wedge (\neg b_1 \vee \neg b_2) \wedge (\neg b_1 \vee \neg b_3)$
- 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-DPLLT(T-formula  $\varphi$ , T-assignment  $\mu$ )
if T-PREPROCESS( $\varphi$ ,  $\mu$ ) = CONFLICT then
    return UNSAT
 $\varphi^P$  ,  $\mu^P \leftarrow \mathbf{T2B}(\varphi)$ ,  $\mathbf{T2B}(\mu)$ 
while (TRUE) do
    T-DECIDE( $\varphi^P$ ,  $\mu^P$ )
    while (TRUE) do
        res  $\leftarrow \mathbf{T-DEDUCE}(\varphi^P, \mu^P)$ 
        if res = SAT then return SAT
        else if res = CONFLICT
            blevel  $\leftarrow \mathbf{T-ANALZECONFLICT}(\varphi^P, \mu^P)$ 
            if (blevel < 0) then return UNSAT
            else T-BACKTRACK(blevel,  $\varphi^P$ ,  $\mu^P$ )
        else break
```

Online DPLL(T)

```
Online-DPLLT(T-formula  $\varphi$ , T-assignment  $\mu$ )
if T-PREPROCESS( $\varphi$ ,  $\mu$ ) = CONFLICT then
    return UNSAT
 $\varphi^P$ ,  $\mu^P \leftarrow \mathbf{T2B}(\varphi)$ ,  $\mathbf{T2B}(\mu)$ 
while (TRUE) do
    T-DECIDE( $\varphi^P$ ,  $\mu^P$ )
    while (TRUE) do
        res  $\leftarrow \mathbf{T-DEDUCE}(\varphi^P, \mu^P)$ 
        if res = SAT then return SAT
        else if res = CONFLICT
            blevel  $\leftarrow \mathbf{T-ANALZECONFLICT}(\varphi^P, \mu^P)$ 
            if (blevel < 0) then return UNSAT
            else T-BACKTRACK(blevel,  $\varphi^P$ ,  $\mu^P$ )
        else break
```

Everything passed by reference.

All procedures have access to T2B and B2T.

Online DPLL(T): T-PREPROCESS

```
Online-DPLLT(T-formula  $\varphi$ , T-assignment  $\mu$ )
if T-PREPROCESS( $\varphi$ ,  $\mu$ ) = CONFLICT then
    return UNSAT
 $\varphi^P$ ,  $\mu^P \leftarrow \mathbf{T2B}(\varphi)$ ,  $\mathbf{T2B}(\mu)$ 
while (TRUE) do
    T-DECIDE( $\varphi^P$ ,  $\mu^P$ )
    while (TRUE) do
        res  $\leftarrow \mathbf{T-DEDUCE}(\varphi^P, \mu^P)$ 
        if res = SAT then return SAT
        else if res = CONFLICT
            blevel  $\leftarrow \mathbf{T-ANALZECONFLICT}(\varphi^P, \mu^P)$ 
            if (blevel < 0) then return UNSAT
            else T-BACKTRACK(blevel,  $\varphi^P$ ,  $\mu^P$ )
        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-DPLLT(T-formula  $\varphi$ , T-assignment  $\mu$ )
if T-PREPROCESS( $\varphi$ ,  $\mu$ ) = CONFLICT then
    return UNSAT
 $\varphi^P$ ,  $\mu^P \leftarrow \text{T2B}(\varphi)$ ,  $\text{T2B}(\mu)$ 
while (TRUE) do
    T-DECIDE( $\varphi^P$ ,  $\mu^P$ )
    while (TRUE) do
        res  $\leftarrow$  T-DEDUCE( $\varphi^P$ ,  $\mu^P$ )
        if res = SAT then return SAT
        else if res = CONFLICT
            blevel  $\leftarrow$  T-ANALZECONFLICT( $\varphi^P$ ,  $\mu^P$ )
            if (blevel < 0) then return UNSAT
            else T-BACKTRACK(blevel,  $\varphi^P$ ,  $\mu^P$ )
        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-DPLLT(T-formula  $\varphi$ , T-assignment  $\mu$ )
if T-PREPROCESS( $\varphi$ ,  $\mu$ ) = CONFLICT then
    return UNSAT
 $\varphi^P$ ,  $\mu^P \leftarrow \mathbf{T2B}(\varphi)$ ,  $\mathbf{T2B}(\mu)$ 
while (TRUE) do
    T-DECIDE( $\varphi^P$ ,  $\mu^P$ )
    while (TRUE) do
        res  $\leftarrow \mathbf{T-DEDUCE}(\varphi^P, \mu^P)$ 
        if res = SAT then return SAT
        else if res = CONFLICT
            blevel  $\leftarrow \mathbf{T-ANALZECONFLICT}(\varphi^P, \mu^P)$ 
            if (blevel < 0) then return UNSAT
            else T-BACKTRACK(blevel,  $\varphi^P$ ,  $\mu^P$ )
        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-DPLLT(T-formula  $\varphi$ , T-assignment  $\mu$ )
if T-PREPROCESS( $\varphi$ ,  $\mu$ ) = CONFLICT then
    return UNSAT
 $\varphi^P$ ,  $\mu^P \leftarrow \text{T2B}(\varphi)$ ,  $\text{T2B}(\mu)$ 
while (TRUE) do
    T-DECIDE( $\varphi^P$ ,  $\mu^P$ )
    while (TRUE) do
        res  $\leftarrow$  T-DEDUCE( $\varphi^P$ ,  $\mu^P$ )
        if res = SAT then return SAT
        else if res = CONFLICT
            blevel  $\leftarrow$  T-ANALZECONFLICT( $\varphi^P$ ,  $\mu^P$ )
            if (blevel < 0) then return UNSAT
            else T-BACKTRACK(blevel,  $\varphi^P$ ,  $\mu^P$ )
        else break
```

Analogous to DECIDE in CDCL:

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

Online DPLL(T): T-DEDUCE

```
Online-DPLLT(T-formula  $\varphi$ , T-assignment  $\mu$ )
if T-PREPROCESS( $\varphi$ ,  $\mu$ ) = CONFLICT then
    return UNSAT
 $\varphi^P$ ,  $\mu^P \leftarrow \mathbf{T2B}(\varphi)$ ,  $\mathbf{T2B}(\mu)$ 
while (TRUE) do
    T-DECIDE( $\varphi^P$ ,  $\mu^P$ )
    while (TRUE) do
        res  $\leftarrow$  T-DEDUCE( $\varphi^P$ ,  $\mu^P$ )
        if res = SAT then return SAT
        else if res = CONFLICT
            blevel  $\leftarrow$  T-ANALZECONFLICT( $\varphi^P$ ,  $\mu^P$ )
            if (blevel < 0) then return UNSAT
            else T-BACKTRACK(blevel,  $\varphi^P$ ,  $\mu^P$ )
        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 $B2T(\mu^P)$ and returns SAT if T-solver does. Otherwise returns CONFLICT.
- no more literals can be deduced: invokes T-solver on partial assignment $B2T(\mu^P)$ and returns CONFLICT if T-solver returns UNSAT. This is *early pruning*. May also do *theory propagation*.

Online DPLL(T): T-DEDUCE

```
Online-DPLLT(T-formula  $\varphi$ , T-assignment  $\mu$ )
if T-PREPROCESS( $\varphi$ ,  $\mu$ ) = CONFLICT then
    return UNSAT
 $\varphi^P$ ,  $\mu^P \leftarrow \mathbf{T2B}(\varphi)$ ,  $\mathbf{T2B}(\mu)$ 
while (TRUE) do
    T-DECIDE( $\varphi^P$ ,  $\mu^P$ )
    while (TRUE) do
        res  $\leftarrow$  T-DEDUCE( $\varphi^P$ ,  $\mu^P$ )
        if res = SAT then return SAT
        else if res = CONFLICT
            blevel  $\leftarrow$  T-ANALZECONFLICT( $\varphi^P$ ,  $\mu^P$ )
            if (blevel < 0) then return UNSAT
            else T-BACKTRACK(blevel,  $\varphi^P$ ,  $\mu^P$ )
        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 $B2T(\mu^P)$ and returns SAT if T-solver does. Otherwise returns CONFLICT.
 - no more literals can be deduced: invokes T-solver on partial assignment $B2T(\mu^P)$ and returns CONFLICT if T-solver returns UNSAT. This is *early pruning*. May also do *theory propagation*.

Online DPLL(T): T-DEDUCE

```
Online-DPLLT(T-formula  $\varphi$ , T-assignment  $\mu$ )
if T-PREPROCESS( $\varphi$ ,  $\mu$ ) = CONFLICT then
    return UNSAT
 $\varphi^P$ ,  $\mu^P \leftarrow \text{T2B}(\varphi)$ ,  $\text{T2B}(\mu)$ 
while (TRUE) do
    T-DECIDE( $\varphi^P$ ,  $\mu^P$ )
    while (TRUE) do
        res  $\leftarrow$  T-DEDUCE( $\varphi^P$ ,  $\mu^P$ )
        if res = SAT then return SAT
        else if res = CONFLICT
            blevel  $\leftarrow$  T-ANALZECONFLICT( $\varphi^P$ ,  $\mu^P$ )
            if (blevel < 0) then return UNSAT
            else T-BACKTRACK(blevel,  $\varphi^P$ ,  $\mu^P$ )
        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 $B2T(\mu^P)$ and returns SAT if T-solver does. Otherwise returns CONFLICT.
 - no more literals can be deduced: invokes T-solver on partial assignment $B2T(\mu^P)$ and returns CONFLICT if T-solver returns UNSAT. This is *early pruning*. May also do *theory propagation*.

Online DPLL(T): T-DEDUCE

```
Online-DPLLT(T-formula  $\varphi$ , T-assignment  $\mu$ )
if T-PREPROCESS( $\varphi$ ,  $\mu$ ) = CONFLICT then
    return UNSAT
 $\varphi^P$ ,  $\mu^P \leftarrow \text{T2B}(\varphi)$ ,  $\text{T2B}(\mu)$ 
while (TRUE) do
    T-DECIDE( $\varphi^P$ ,  $\mu^P$ )
    while (TRUE) do
        res  $\leftarrow$  T-DEDUCE( $\varphi^P$ ,  $\mu^P$ )
        if res = SAT then return SAT
        else if res = CONFLICT
            blevel  $\leftarrow$  T-ANALZECONFLICT( $\varphi^P$ ,  $\mu^P$ )
            if (blevel < 0) then return UNSAT
            else T-BACKTRACK(blevel,  $\varphi^P$ ,  $\mu^P$ )
        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 $B2T(\mu^P)$ and returns SAT if T-solver does. Otherwise returns CONFLICT.
 - no more literals can be deduced: invokes T-solver on partial assignment $B2T(\mu^P)$ and returns CONFLICT if T-solver returns UNSAT. This is *early pruning*. May also do *theory propagation*.

Online DPLL(T): T-ANALYZECONFLICT

```
Online-DPLLT(T-formula  $\varphi$ , T-assignment  $\mu$ )
if T-PREPROCESS( $\varphi$ ,  $\mu$ ) = CONFLICT then
    return UNSAT
 $\varphi^P$ ,  $\mu^P \leftarrow \mathbf{T2B}(\varphi)$ ,  $\mathbf{T2B}(\mu)$ 
while (TRUE) do
    T-DECIDE( $\varphi^P$ ,  $\mu^P$ )
    while (TRUE) do
        res  $\leftarrow$  T-DEDUCE( $\varphi^P$ ,  $\mu^P$ )
        if res = SAT then return SAT
        else if res = CONFLICT
            blevel  $\leftarrow$  T-ANALZECONFLICT( $\varphi^P$ ,  $\mu^P$ )
            if (blevel < 0) then return UNSAT
            else T-BACKTRACK(blevel,  $\varphi^P$ ,  $\mu^P$ )
        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-DPLLT(T-formula  $\varphi$ , T-assignment  $\mu$ )
if T-PREPROCESS( $\varphi$ ,  $\mu$ ) = CONFLICT then
    return UNSAT
 $\varphi^P$ ,  $\mu^P$   $\leftarrow$  T2B( $\varphi$ ), T2B( $\mu$ )
while (TRUE) do
    T-DECIDE( $\varphi^P$ ,  $\mu^P$ )
    while (TRUE) do
        res  $\leftarrow$  T-DEDUCE( $\varphi^P$ ,  $\mu^P$ )
        if res = SAT then return SAT
        else if res = CONFLICT
            blevel  $\leftarrow$  T-ANALZECONFLICT( $\varphi^P$ ,  $\mu^P$ )
            if (blevel < 0) then return UNSAT
            else T-BACKTRACK(blevel,  $\varphi^P$ ,  $\mu^P$ )
        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-DPLLT(T-formula  $\varphi$ , T-assignment  $\mu$ )
if T-PREPROCESS( $\varphi$ ,  $\mu$ ) = CONFLICT then
    return UNSAT
 $\varphi^P$ ,  $\mu^P$   $\leftarrow$  T2B( $\varphi$ ), T2B( $\mu$ )
while (TRUE) do
    T-DECIDE( $\varphi^P$ ,  $\mu^P$ )
    while (TRUE) do
        res  $\leftarrow$  T-DEDUCE( $\varphi^P$ ,  $\mu^P$ )
        if res = SAT then return SAT
        else if res = CONFLICT
            blevel  $\leftarrow$  T-ANALZECONFLICT( $\varphi^P$ ,  $\mu^P$ )
            if (blevel < 0) then return UNSAT
            else T-BACKTRACK(blevel,  $\varphi^P$ ,  $\mu^P$ )
        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-DPLLT(T-formula  $\varphi$ , T-assignment  $\mu$ )
if T-PREPROCESS( $\varphi$ ,  $\mu$ ) = CONFLICT then
    return UNSAT
 $\varphi^P$ ,  $\mu^P$   $\leftarrow$  T2B( $\varphi$ ), T2B( $\mu$ )
while (TRUE) do
    T-DECIDE( $\varphi^P$ ,  $\mu^P$ )
    while (TRUE) do
        res  $\leftarrow$  T-DEDUCE( $\varphi^P$ ,  $\mu^P$ )
        if res = SAT then return SAT
        else if res = CONFLICT
            blevel  $\leftarrow$  T-ANALZECONFLICT( $\varphi^P$ ,  $\mu^P$ )
            if (blevel < 0) then return UNSAT
            else T-BACKTRACK(blevel,  $\varphi^P$ ,  $\mu^P$ )
        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-DPLLT(T-formula  $\varphi$ , T-assignment  $\mu$ )
if T-PREPROCESS( $\varphi$ ,  $\mu$ ) = CONFLICT then
    return UNSAT
 $\varphi^P$ ,  $\mu^P \leftarrow \mathbf{T2B}(\varphi)$ ,  $\mathbf{T2B}(\mu)$ 
while (TRUE) do
    T-DECIDE( $\varphi^P$ ,  $\mu^P$ )
    while (TRUE) do
        res  $\leftarrow$  T-DEDUCE( $\varphi^P$ ,  $\mu^P$ )
        if res = SAT then return SAT
        else if res = CONFLICT
            blevel  $\leftarrow$  T-ANALZECONFLICT( $\varphi^P$ ,  $\mu^P$ )
            if (blevel < 0) then return UNSAT
            else T-BACKTRACK(blevel,  $\varphi^P$ ,  $\mu^P$ )
        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

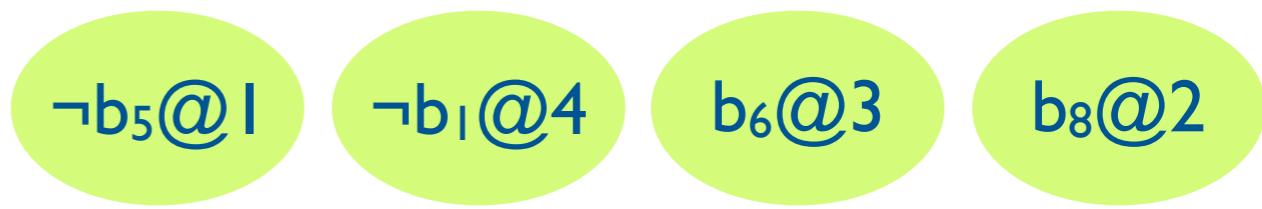
T_R -formula φ and φ^P :

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. $A_1 \vee (x_3 = 3x_5 + 4) \vee A_2, A_1 \vee b_8 \vee A_2$

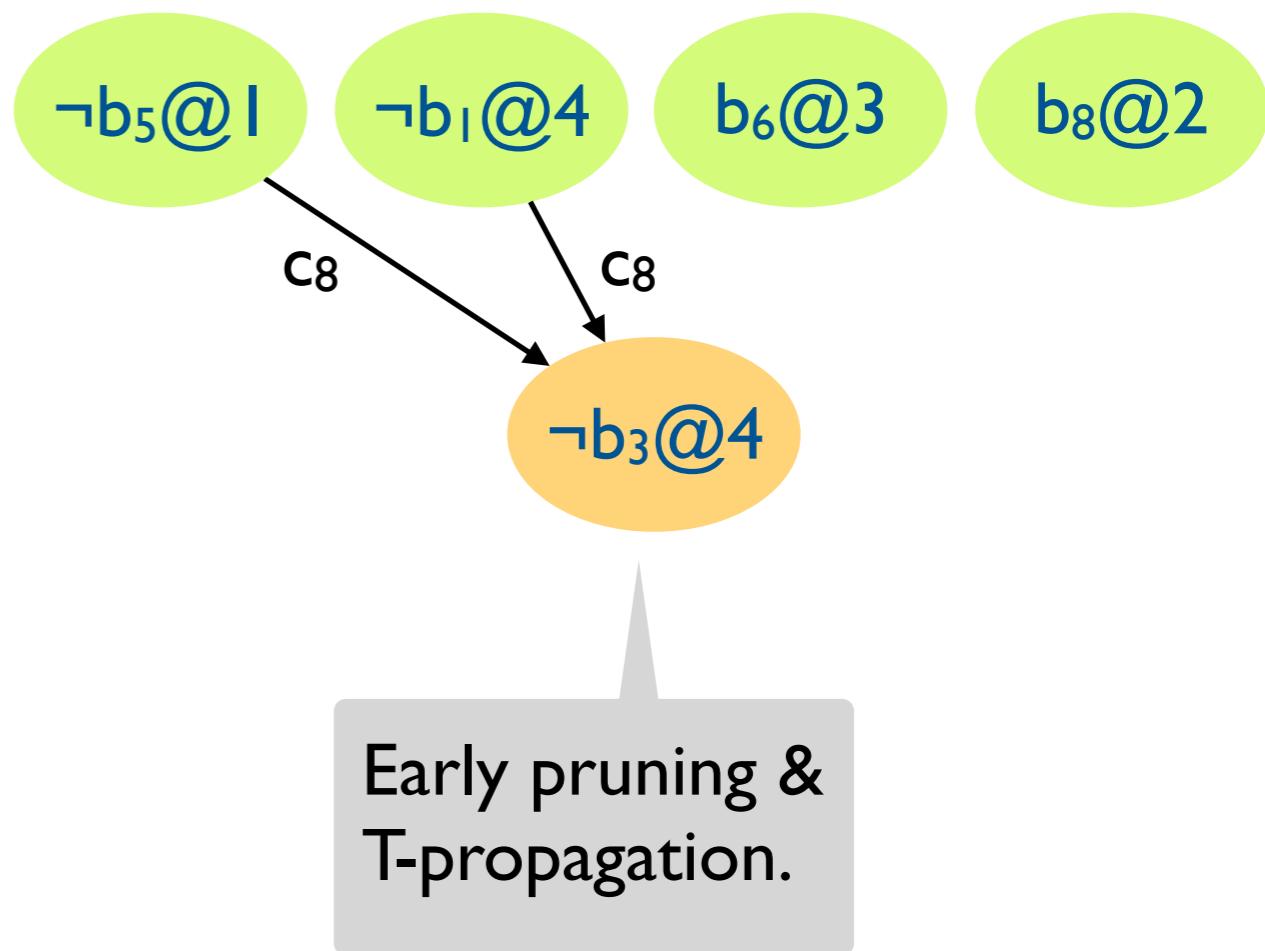


T-DECIDE makes 4 decisions.

Online DPLL(T) example: T-Deduce

T_R -formula φ and φ^P :

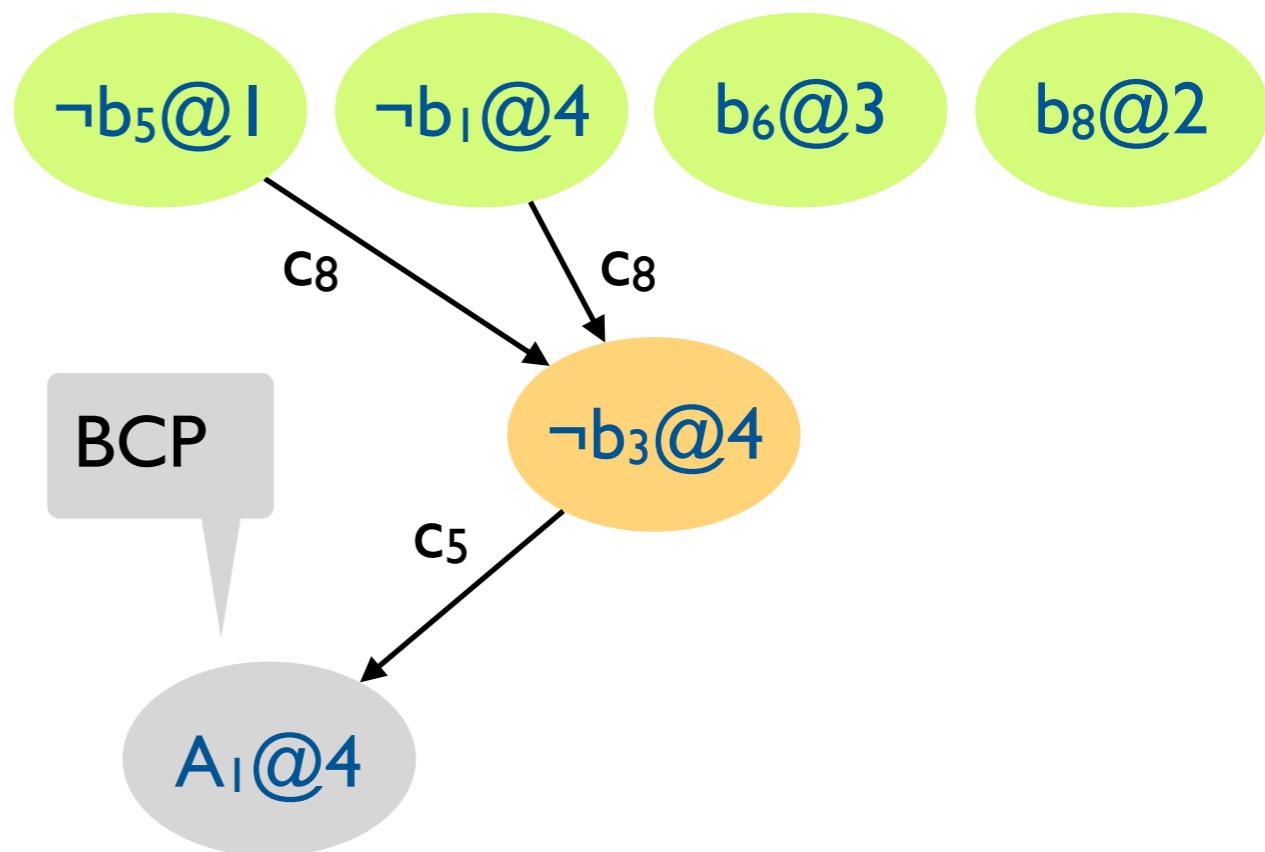
1. $\neg(2x_2 - x_3 > 2) \vee A_1, \neg b_1 \vee A_1$
2. $\neg A_2 \vee (x_1 - x_5 \leq 1), \neg A_2 \vee b_2$
3. $(3x_1 - 2x_2 \leq 3) \vee A_2, b_3 \vee A_2$
4. $\neg(2x_3 + x_4 \geq 5) \vee \neg(3x_1 - x_3 \leq 6) \vee \neg A_1, \neg b_4 \vee \neg b_5 \vee \neg A_1$
5. $A_1 \vee (3x_1 - 2x_2 \leq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



Online DPLL(T) example: T-Deduce

T_R -formula φ and φ^P :

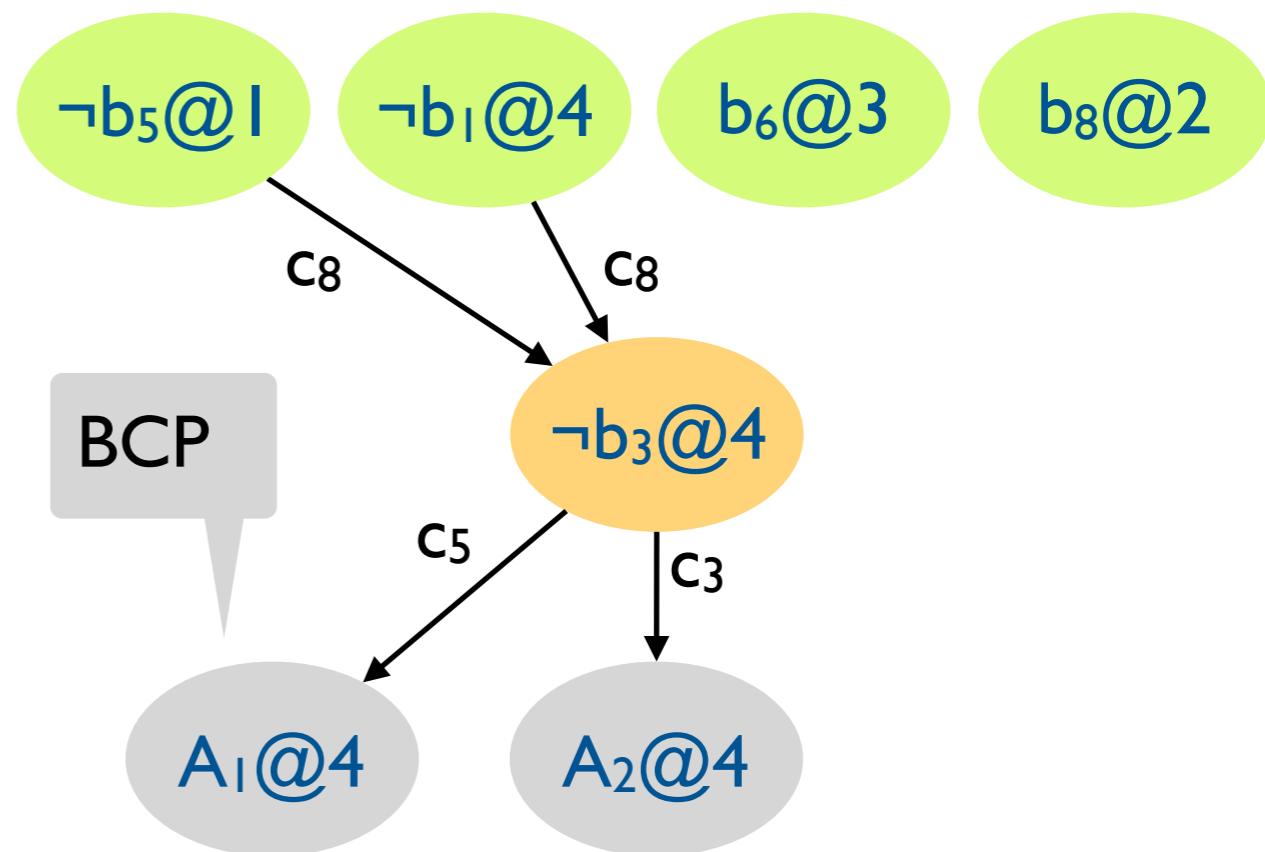
1. $\neg(2x_2 - x_3 > 2) \vee A_1, \neg b_1 \vee A_1$
2. $\neg A_2 \vee (x_1 - x_5 \leq 1), \neg A_2 \vee b_2$
3. $(3x_1 - 2x_2 \leq 3) \vee A_2, b_3 \vee A_2$
4. $\neg(2x_3 + x_4 \geq 5) \vee \neg(3x_1 - x_3 \leq 6) \vee \neg A_1, \neg b_4 \vee \neg b_5 \vee \neg A_1$
5. $A_1 \vee (3x_1 - 2x_2 \leq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

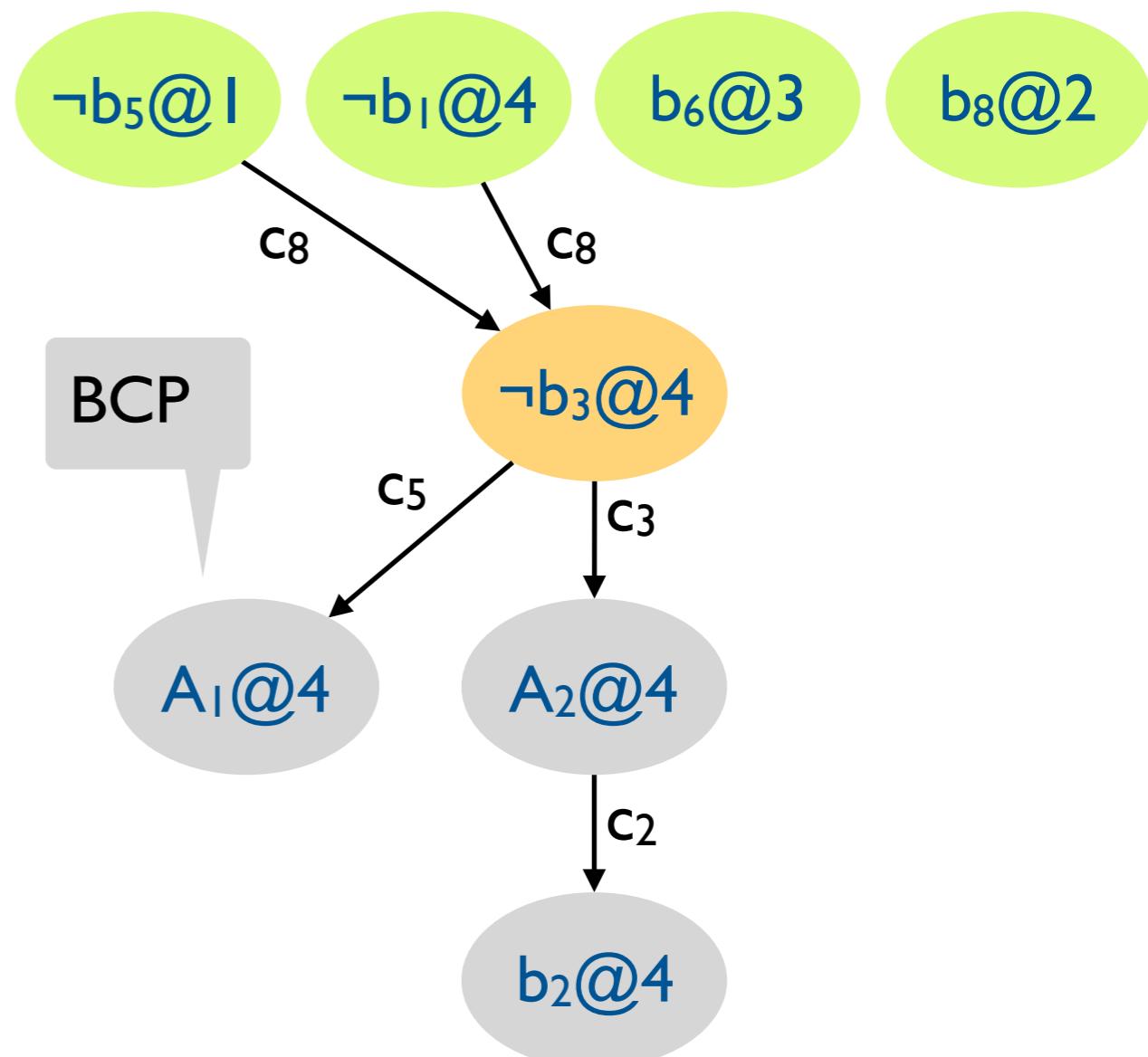
1. $\neg(2x_2 - x_3 > 2) \vee A_1, \neg b_1 \vee A_1$
2. $\neg A_2 \vee (x_1 - x_5 \leq 1), \neg A_2 \vee b_2$
3. $(3x_1 - 2x_2 \leq 3) \vee A_2, b_3 \vee A_2$
4. $\neg(2x_3 + x_4 \geq 5) \vee \neg(3x_1 - x_3 \leq 6) \vee \neg A_1, \neg b_4 \vee \neg b_5 \vee \neg A_1$
5. $A_1 \vee (3x_1 - 2x_2 \leq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



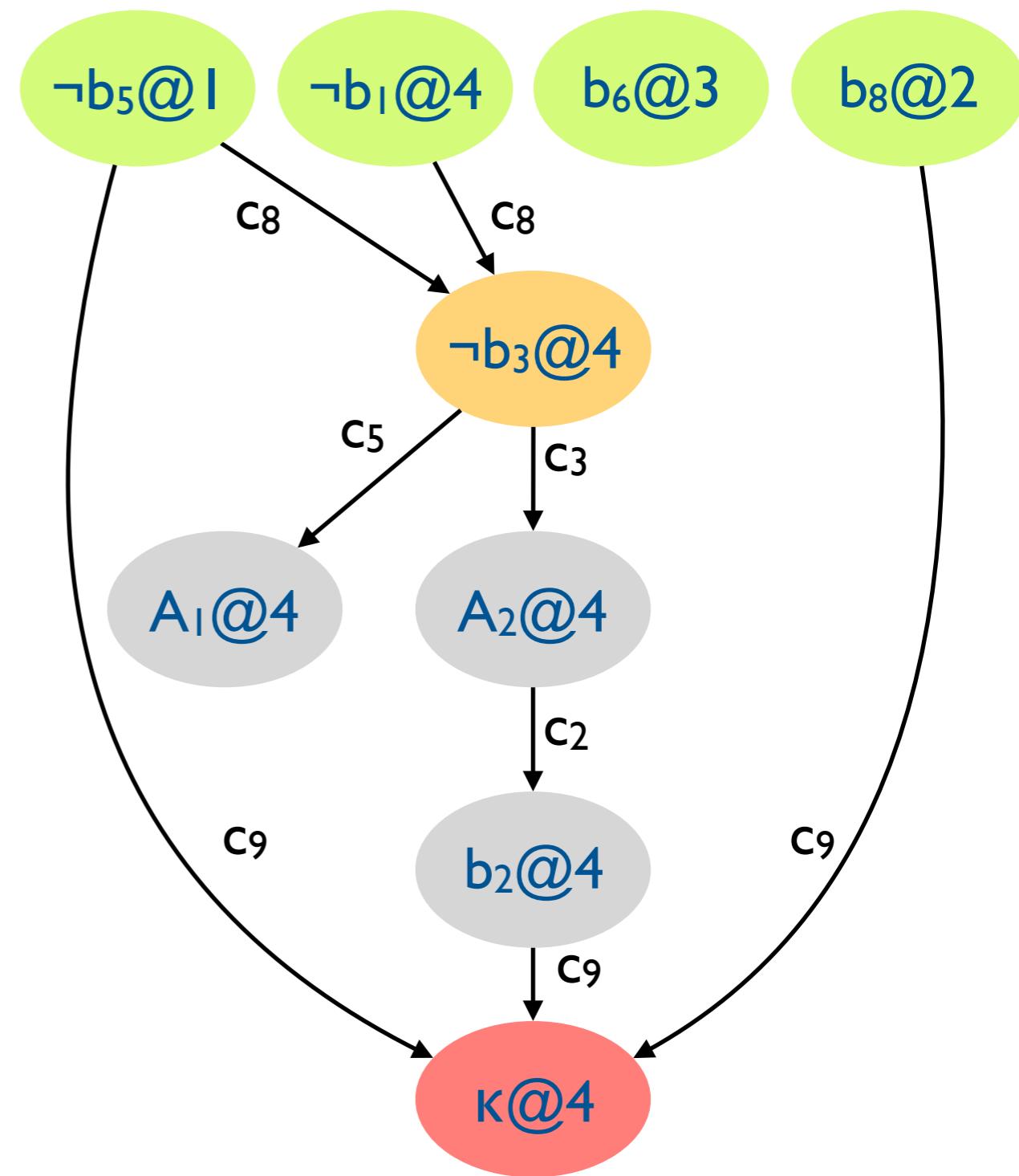
Online DPLL(T) example: T-Deduce

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.

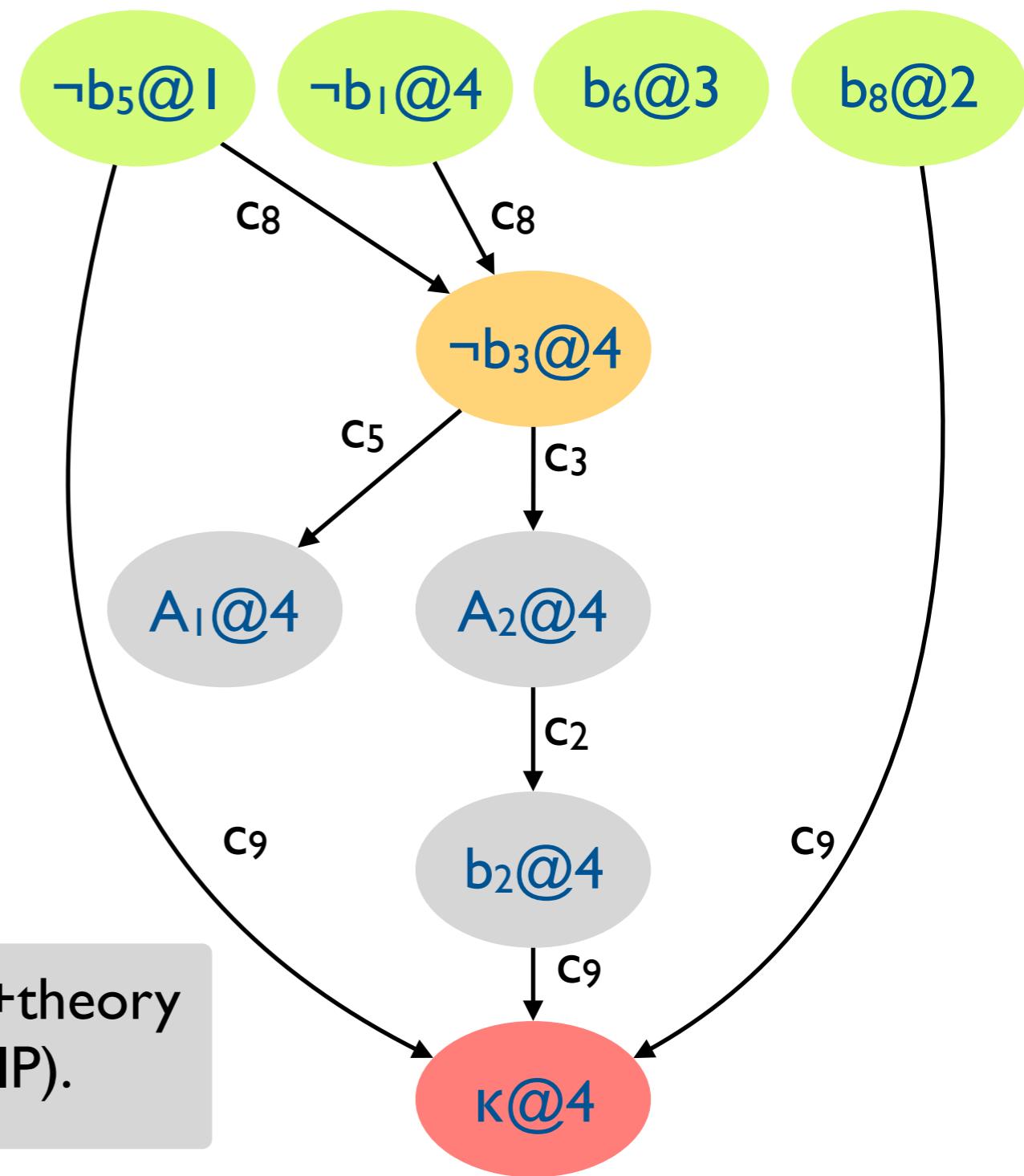


Online DPLL(T) example: T-AnalyzeConflict

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).

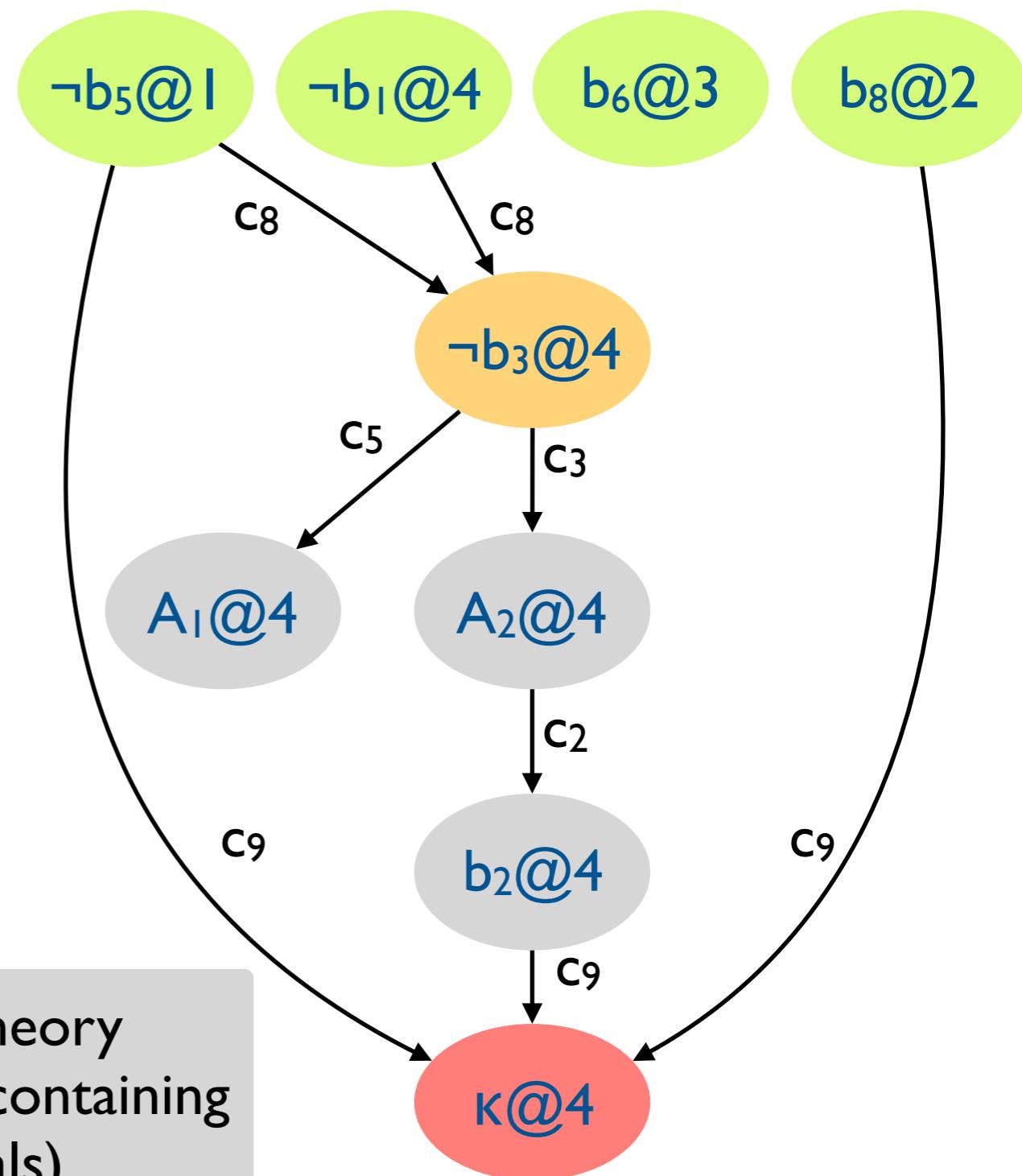


Online DPLL(T) example: T-AnalyzeConflict

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. $\neg(2x_2 - x_3 > 2) \vee A_1, \neg b_1 \vee A_1$ ¬b₅@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. $b_5 \vee \neg b_8 \vee b_1$

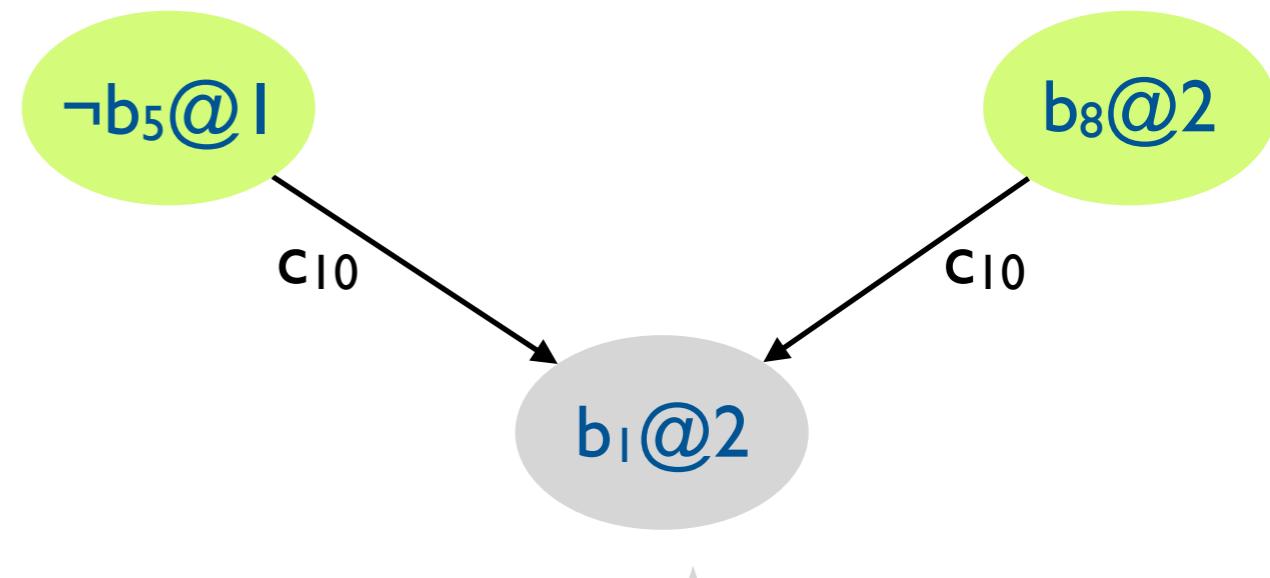
b₈@2

Backtrack to the second highest decision level (2).

Online DPLL(T) example: T-Backtrack

T_R -formula φ and φ^P :

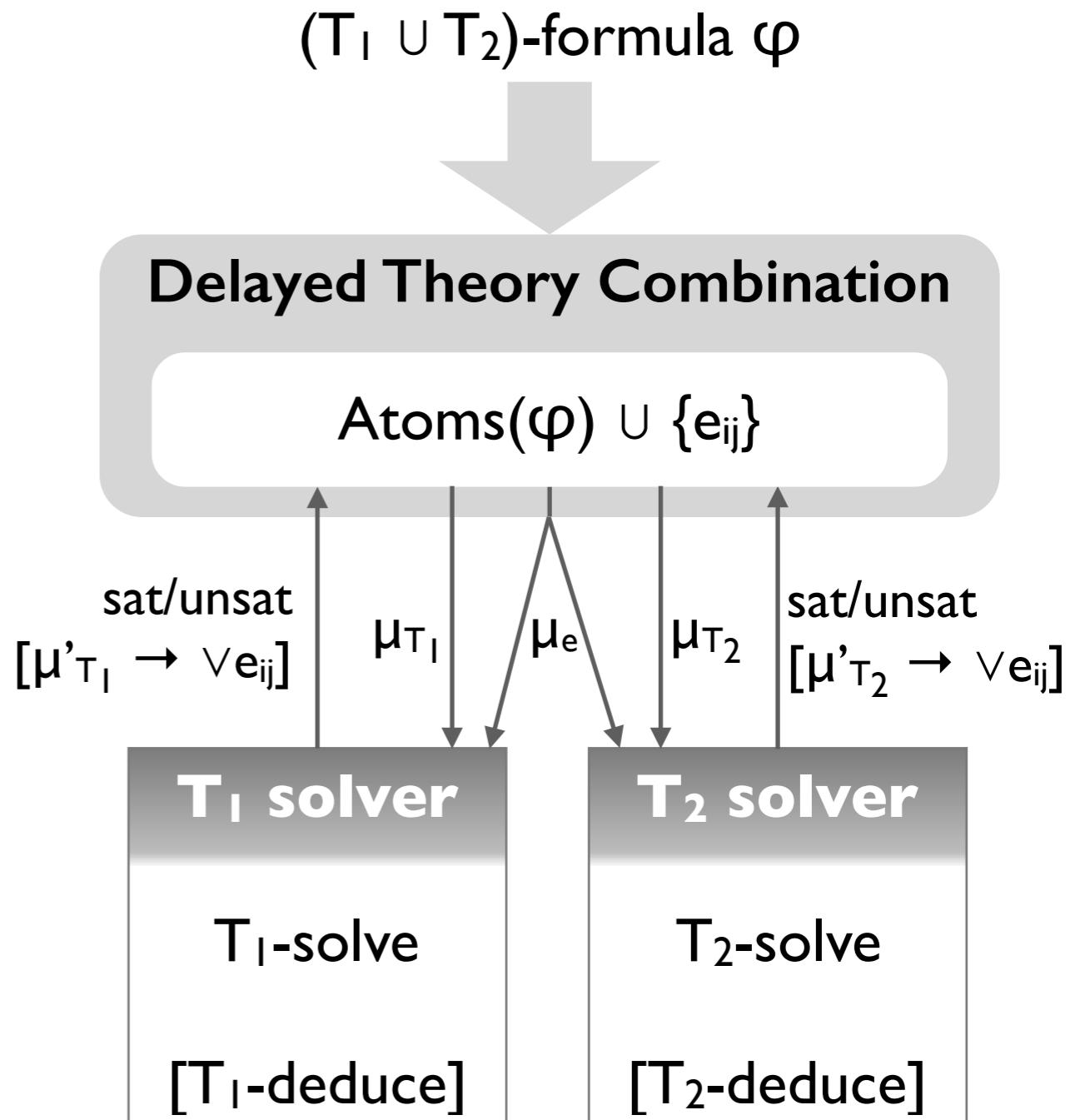
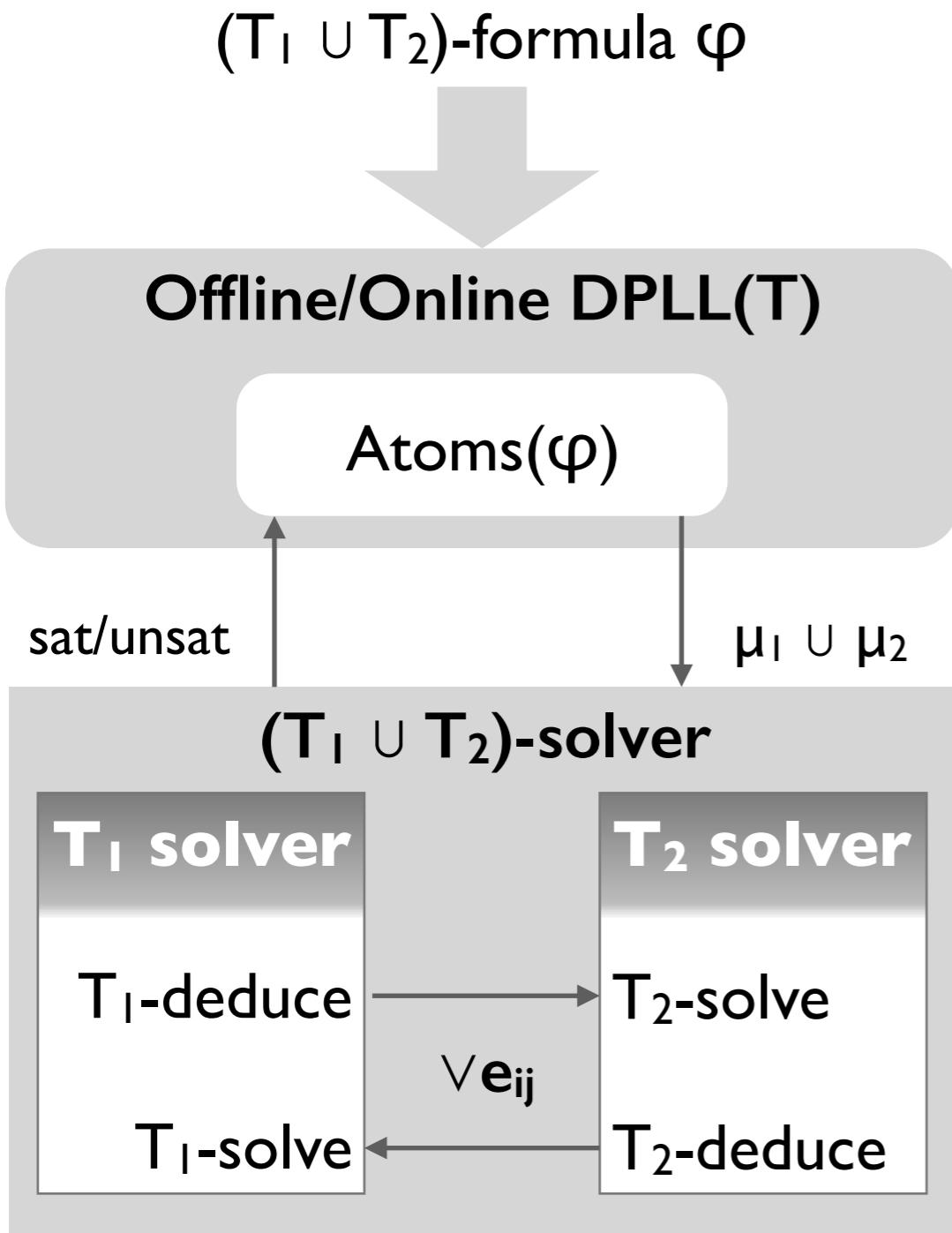
1. $\neg(2x_2 - x_3 > 2) \vee A_1, \neg b_1 \vee A_1$
2. $\neg A_2 \vee (x_1 - x_5 \leq 1), \neg A_2 \vee b_2$
3. $(3x_1 - 2x_2 \leq 3) \vee A_2, b_3 \vee A_2$
4. $\neg(2x_3 + x_4 \geq 5) \vee \neg(3x_1 - x_3 \leq 6) \vee \neg A_1, \neg b_4 \vee \neg b_5 \vee \neg A_1$
5. $A_1 \vee (3x_1 - 2x_2 \leq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



b_1 immediately implied by BCP.

Backtrack to the second highest decision level (2).

The DPLL(T) Framework



Delayed Theory Combination (DTC)

```
Online-DPLLT(T-formula  $\varphi$ , T-assignment  $\mu$ )
if T-PREPROCESS( $\varphi, \mu$ ) = CONFLICT then
    return UNSAT
 $\varphi^P, \mu^P \leftarrow \mathbf{T2B}(\varphi), \mathbf{T2B}(\mu)$ 
while (TRUE) do
    T-DECIDE( $\varphi^P, \mu^P$ )
    while (TRUE) do
        res  $\leftarrow$  T-DEDUCE( $\varphi^P, \mu^P$ )
        if res = SAT then return SAT
        else if res = CONFLICT
            blevel  $\leftarrow$  T-ANALZECONFLICT( $\varphi^P, \mu^P$ )
            if (blevel < 0) then return UNSAT
            else T-BACKTRACK(blevel,  $\varphi^P, \mu^P$ )
        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-ANALZECONFLICT 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!