A Verified SAT Solver Framework

Jasmin C. Blanchette
Mathias Fleury
Christoph Weidenbach

SAT Solving

Given a formula in conjunctive normal form

$$\varphi = \bigwedge_i \bigvee_j L_{i,j}$$

is there an assignment making the formula true?

Most used algorithm: CDCL, an improvement over DPLL

SAT has many applications

nature International weekly journal of science

Home | News & Comment | Research | Careers & Jobs | Current Issue | Archive | Audio & Video | For

Archive > Volume 534 > Issue 7605 > News > Article

NATURE | NEWS

Two-hundred-terabyte maths proof is largest ever

(Wednesday: “Solving Very Hard Problems: Cube-and-Conquer, a Hybrid SAT Solving Method”)

How reliable are SAT solvers?

Two ways to ensure correctness:

- certify the certificate
  - certificates are huge
- verification of the code
  - code will not be competitive
  - allows to study metatheory

INVENTORS FOR THE DIGITAL WORLD

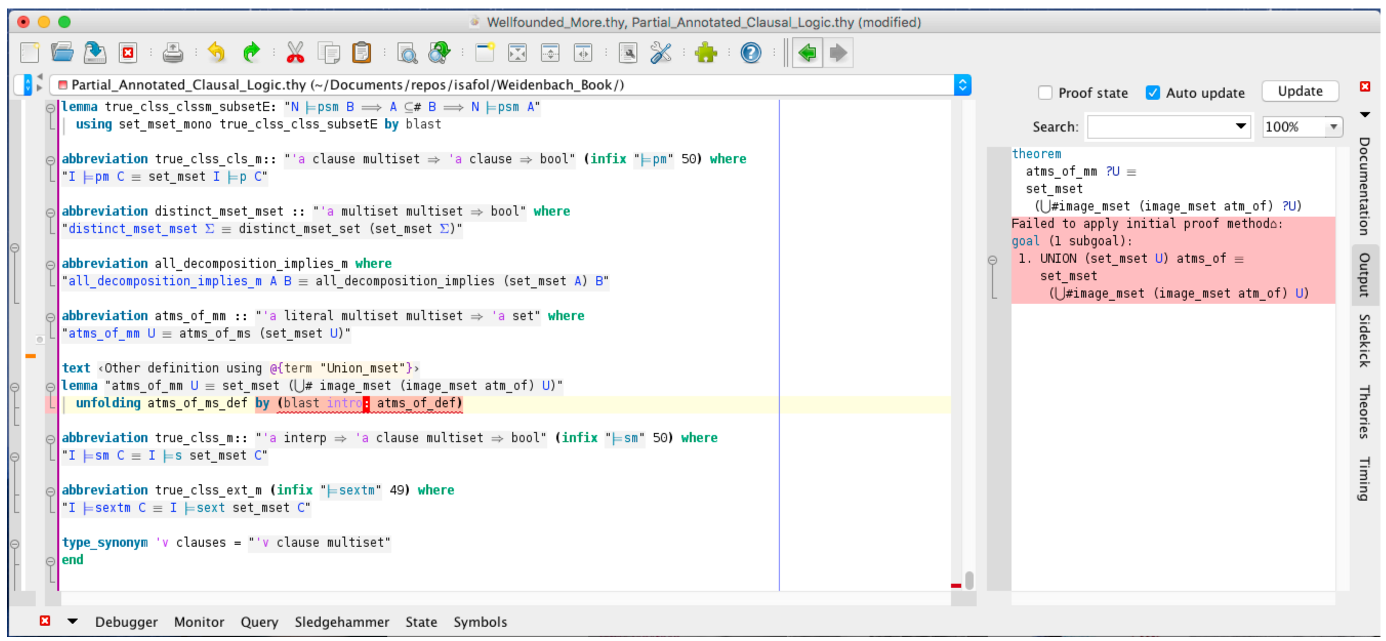
VRIJE UNIVERSITEIT AMSTERDAM

max planck institut informatik

	Correctness	Applicability
Theory behind SAT solvers	Proof	*every* input
Run of a SAT solver	Certificate: proof of (un)satisfiability	*a given* input

Theorem proving: Interactive vs Automated

Interactive



Automated

```
$ minisat eq.atree.braun.7.unsat.cnf
  UNSATISFIABLE
```

```
$ minisat eq.atree.braun.8.unsat.cnf
  UNKNOWN
```

I certify your proof

IsaFoL project

Isabelle Formalisation of Logic

IsaFoL

- FO resolution
  by Schlichtkrull (ITP 2016)

- CDCL with learn, forget, restart, and incrementality
  by Blanchette, Fleury, Weidenbach (IJCAR 2016, now)

- FO ordered resolution with selection
  by Blanchette, Schlichtkrull, Traytel (ongoing)

- GRAT certificate checker
  by Lammich (CADE-26, 2017)

IsaFoL

- FO resolution
  by Schlichtkrull (ITP 2016)

- CDCL with learn, forget, restart, and incrementality
  by Blanchette, Fleury, Weidenbach (IJCAR 2016, now)

- FO ordered resolution with selection
  by Blanchette, Schlichtkrull, Traytel (ongoing)

- GRAT certificate checker
  by Lammich (CADE-26, 2017)

Inria INVENTORS FOR THE DIGITAL WORLD VU VRIJE UNIVERSITEIT AMSTERDAM max planck institut informatik

- Eat our own dog food

  case study for proof assistants and automatic provers

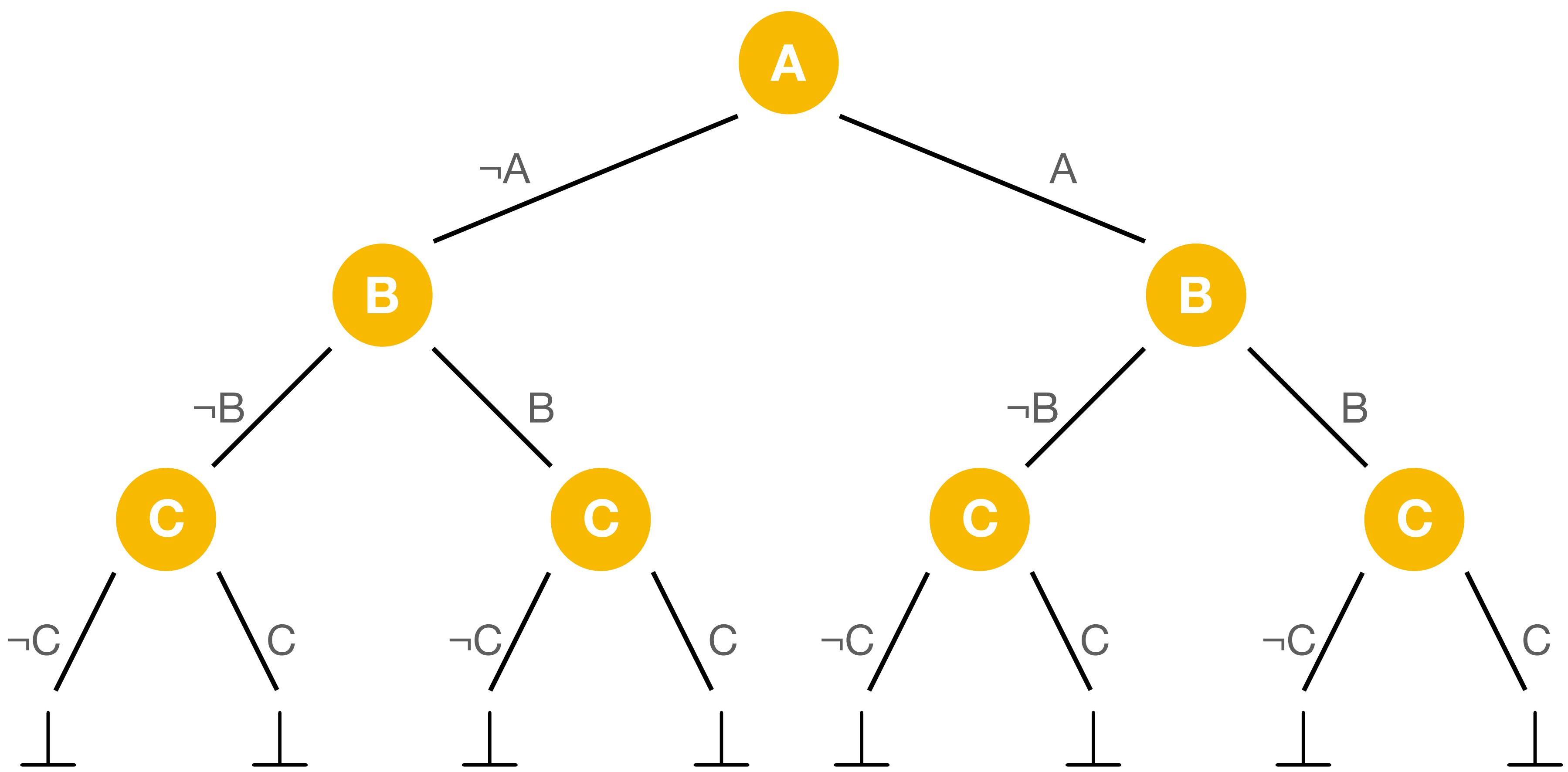
- Build libraries for state-of-the-art research

  Automated Reasoning:
  The Art of Generic Problem Solving
  (forthcoming textbook by Weidenbach)

Truth Table

N = $A \vee B \vee C \quad \neg A \vee B \vee C \quad \neg B \vee C \quad B \vee \neg C$

$\neg A \vee B \quad A \vee \neg B \vee \neg C \quad A \vee \neg C$

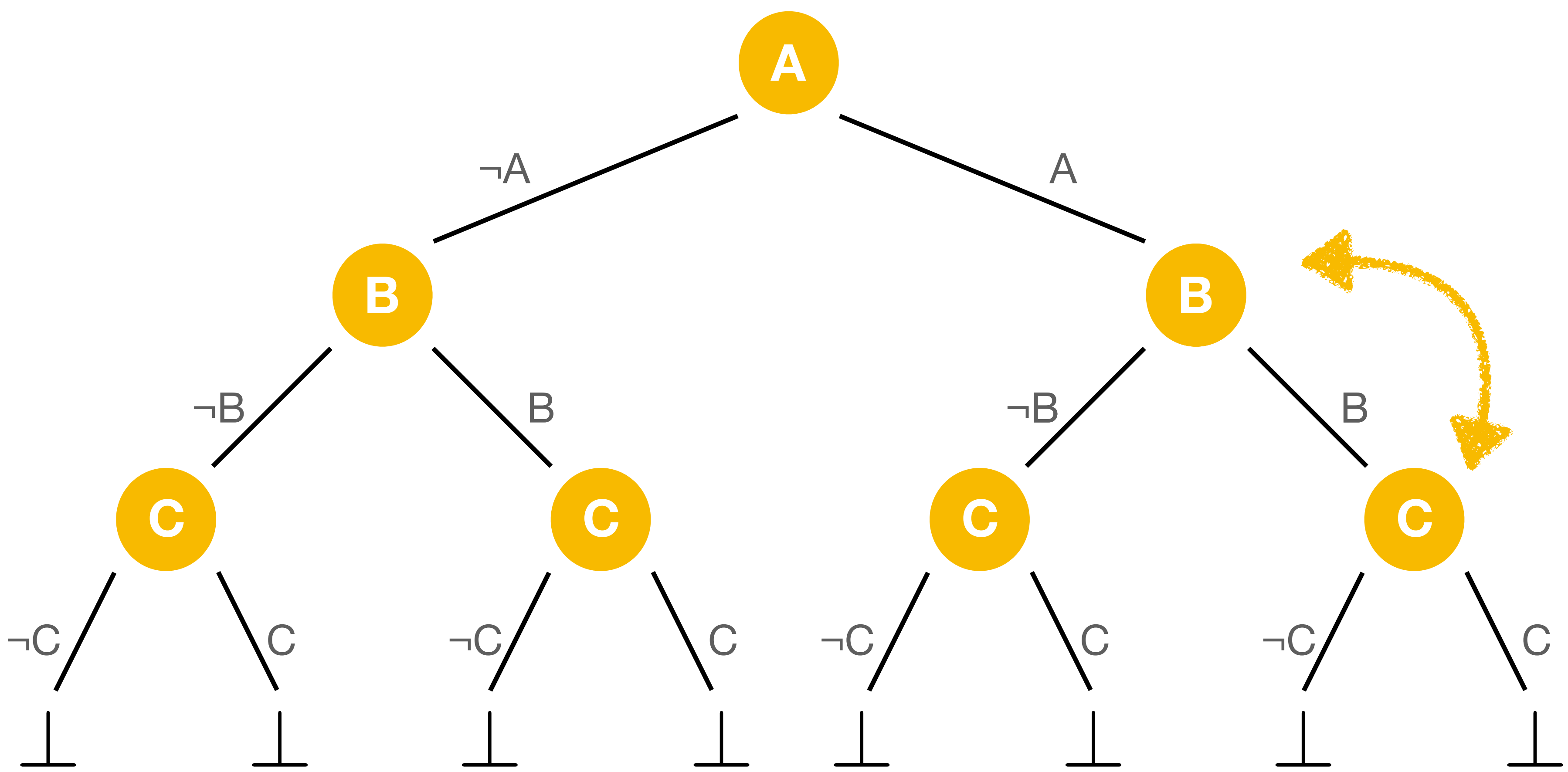


Decide

Truth Table

N = $A \vee B \vee C \quad \neg A \vee B \vee C \quad \neg B \vee C \quad B \vee \neg C$

$\neg A \vee B \quad A \vee \neg B \vee \neg C \quad A \vee \neg C$



Decide

DPLL

N = $A \vee B \vee C \quad \neg A \vee B \vee C \quad \neg B \vee C \quad B \vee \neg C$

$\neg A \vee B \quad A \vee \neg B \vee \neg C \quad A \vee \neg C$

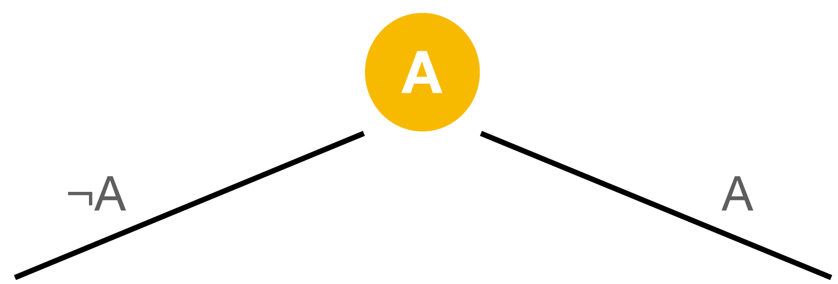
Decide

Propagate

DPLL

N = $A \vee B \vee C \quad \neg A \vee B \vee C \quad \neg B \vee C \quad B \vee \neg C$

$\neg A \vee B \quad A \vee \neg B \vee \neg C \quad A \vee \neg C$



Decide

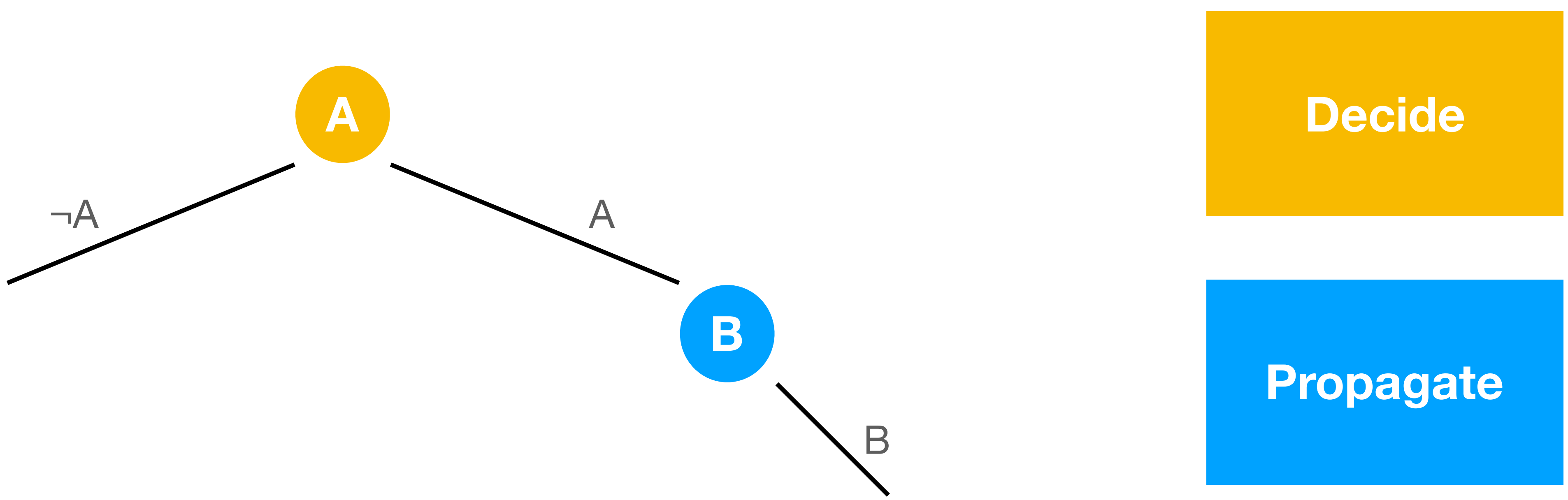
Propagate

DPLL

N =

$A \vee B \vee C \quad \neg A \vee B \vee C \quad \neg B \vee C \quad B \vee \neg C$

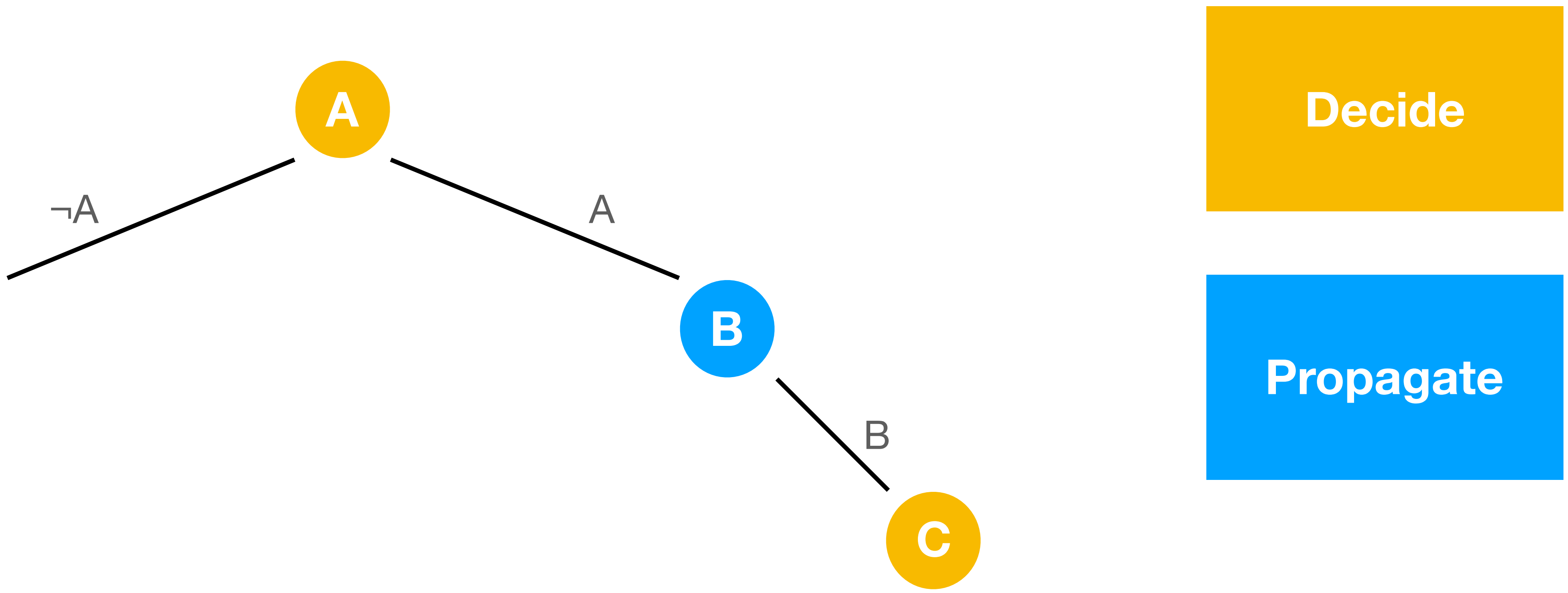
$\neg A \vee B \quad A \vee \neg B \vee \neg C \quad A \vee \neg C$



DPLL

N = $A \vee B \vee C \quad \neg A \vee B \vee C \quad \neg B \vee C \quad B \vee \neg C$

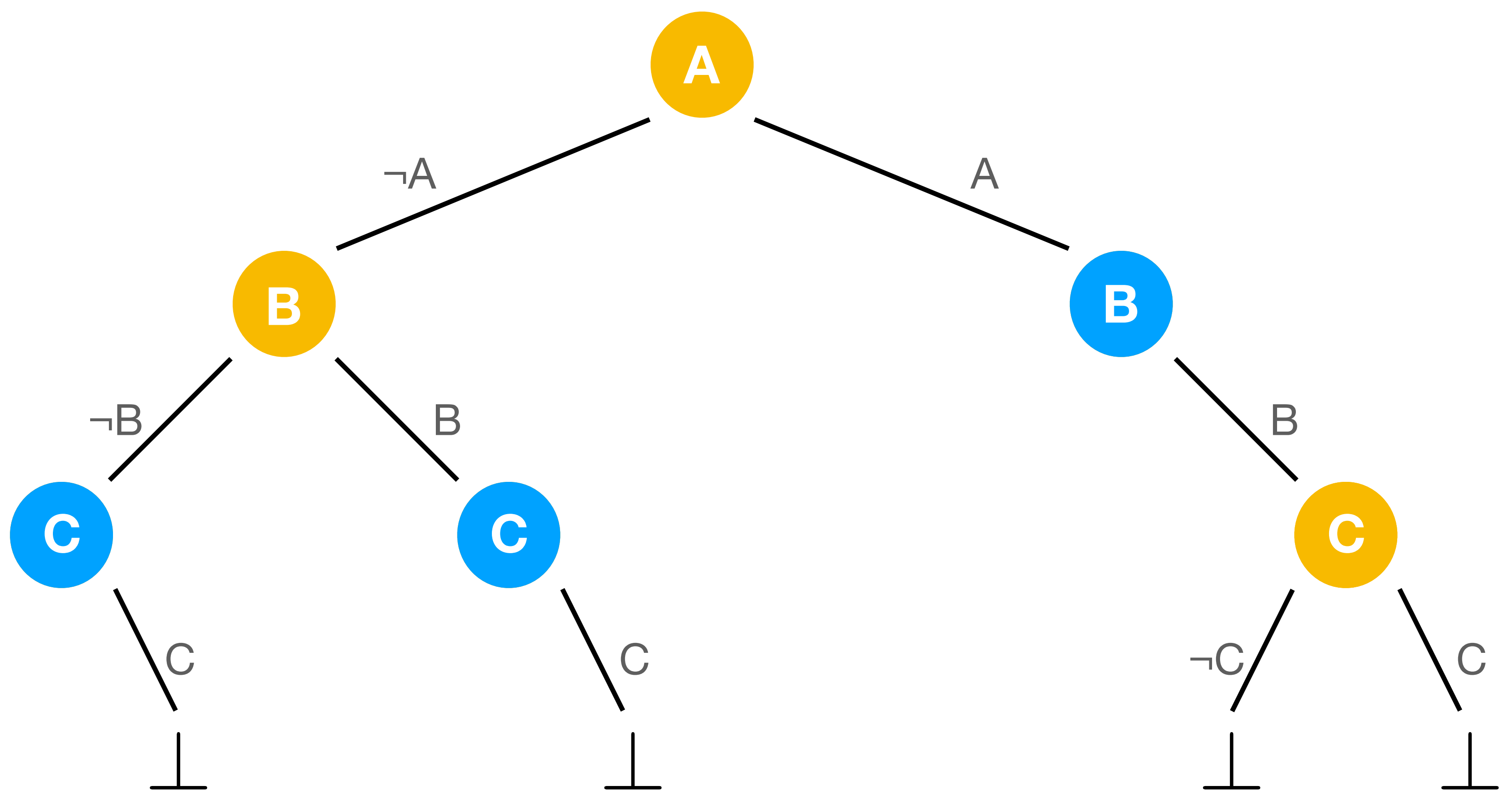
$\neg A \vee B \quad A \vee \neg B \vee \neg C \quad A \vee \neg C$



DPLL

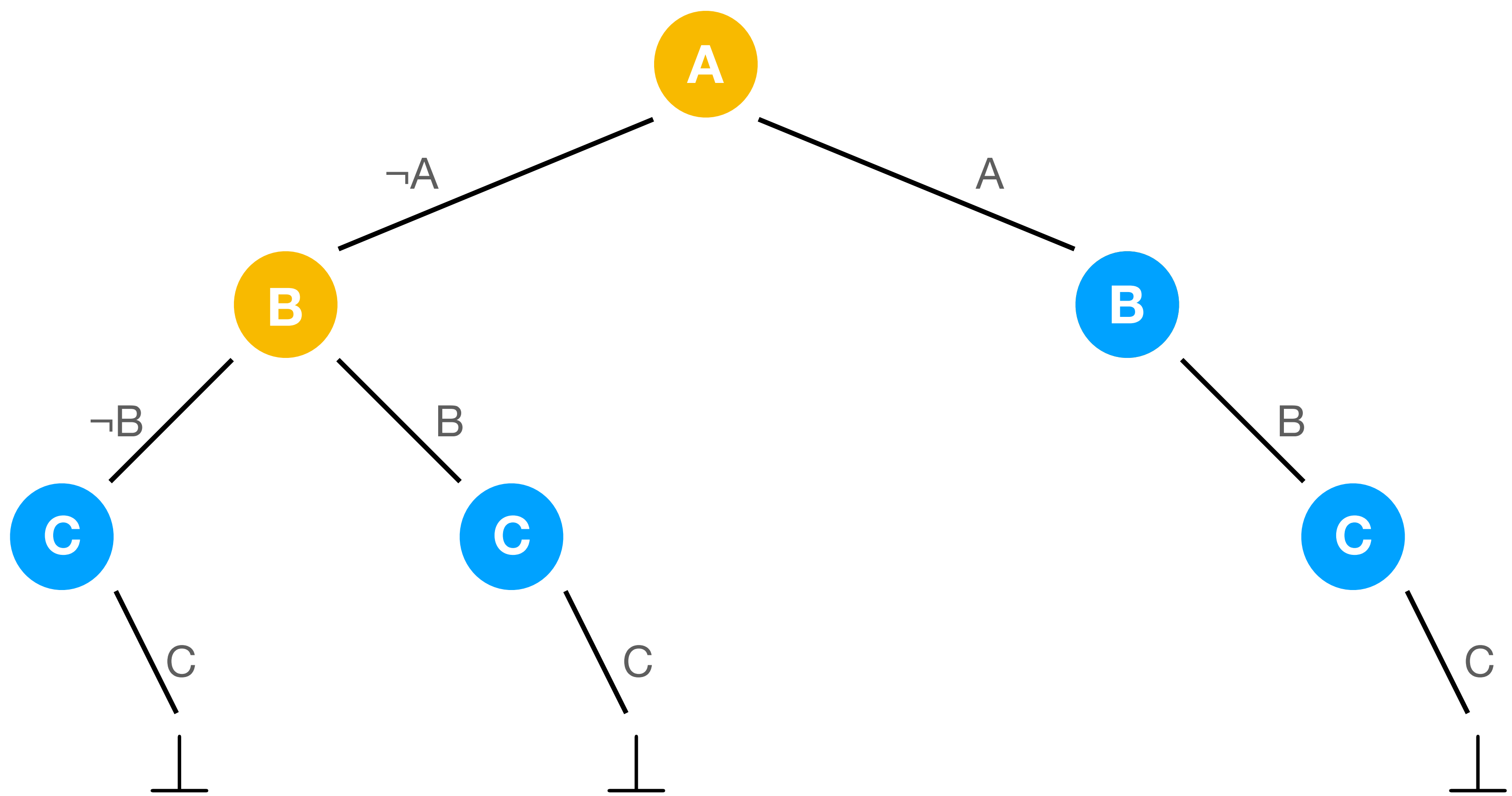
N = $A \vee B \vee C \quad \neg A \vee B \vee C \quad \neg B \vee C \quad B \vee \neg C$

$\neg A \vee B \quad A \vee \neg B \vee \neg C \quad A \vee \neg C$



Decide

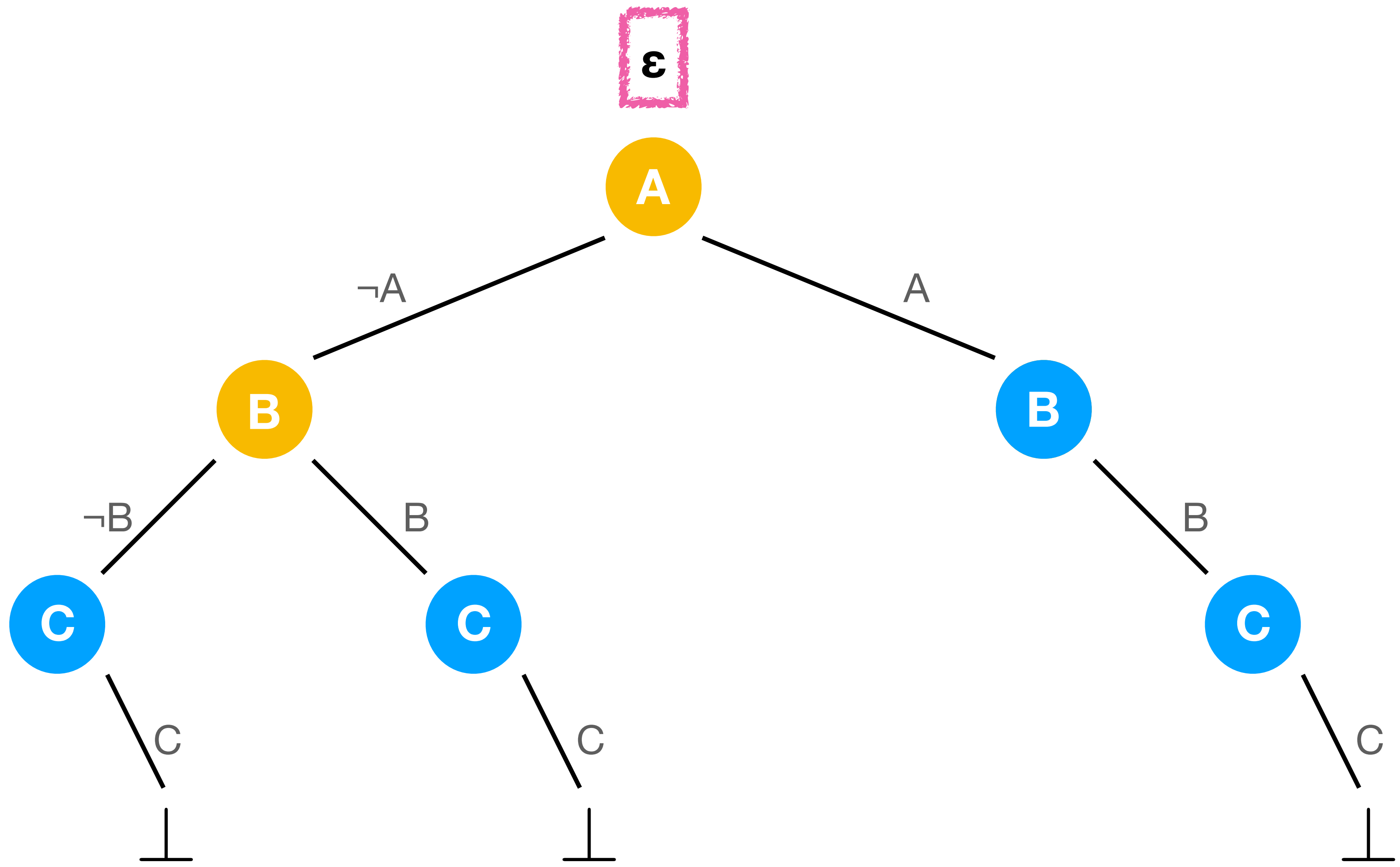
Propagate



State in Isabelle

Decide

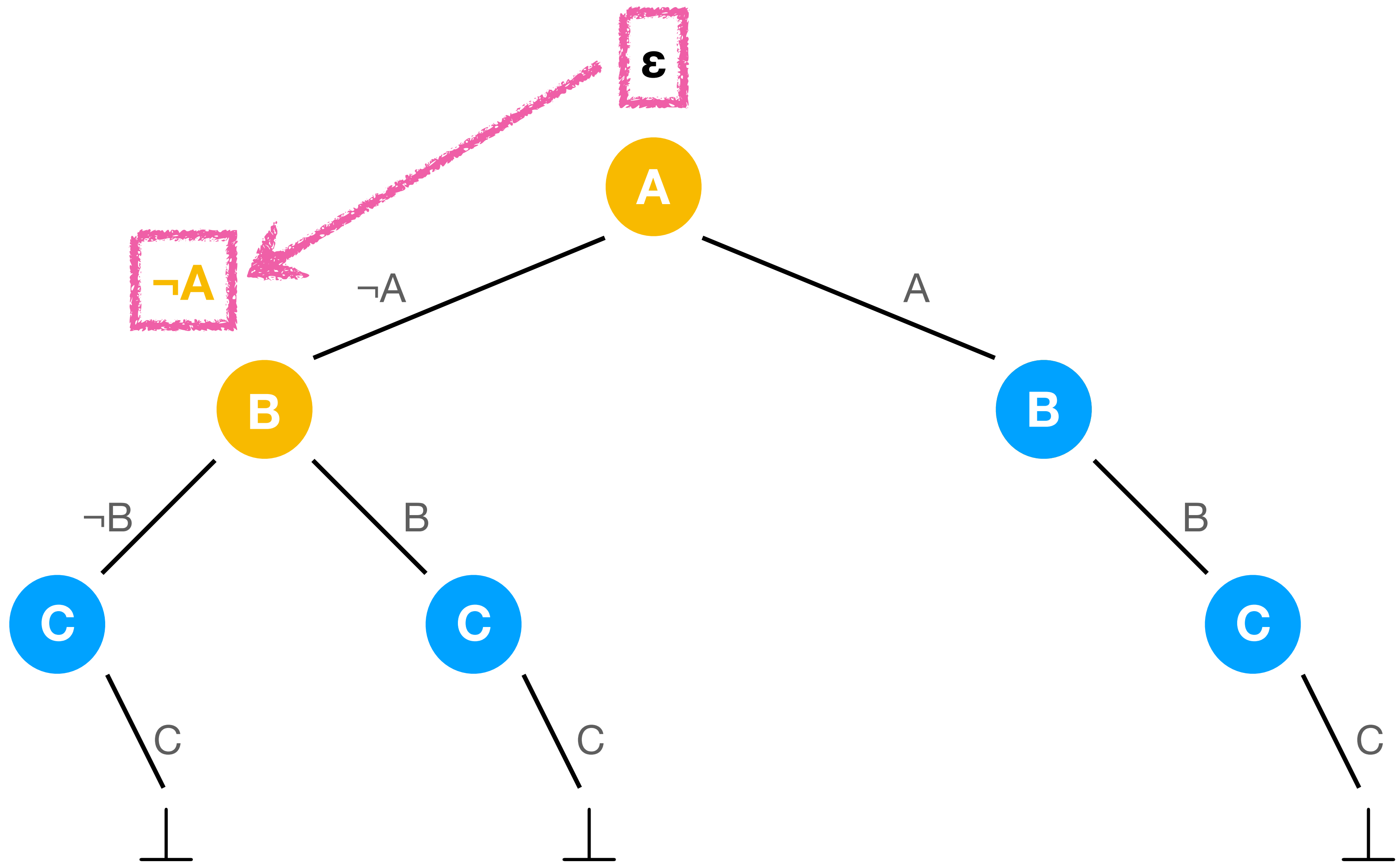
Propagate



State in Isabelle

Decide

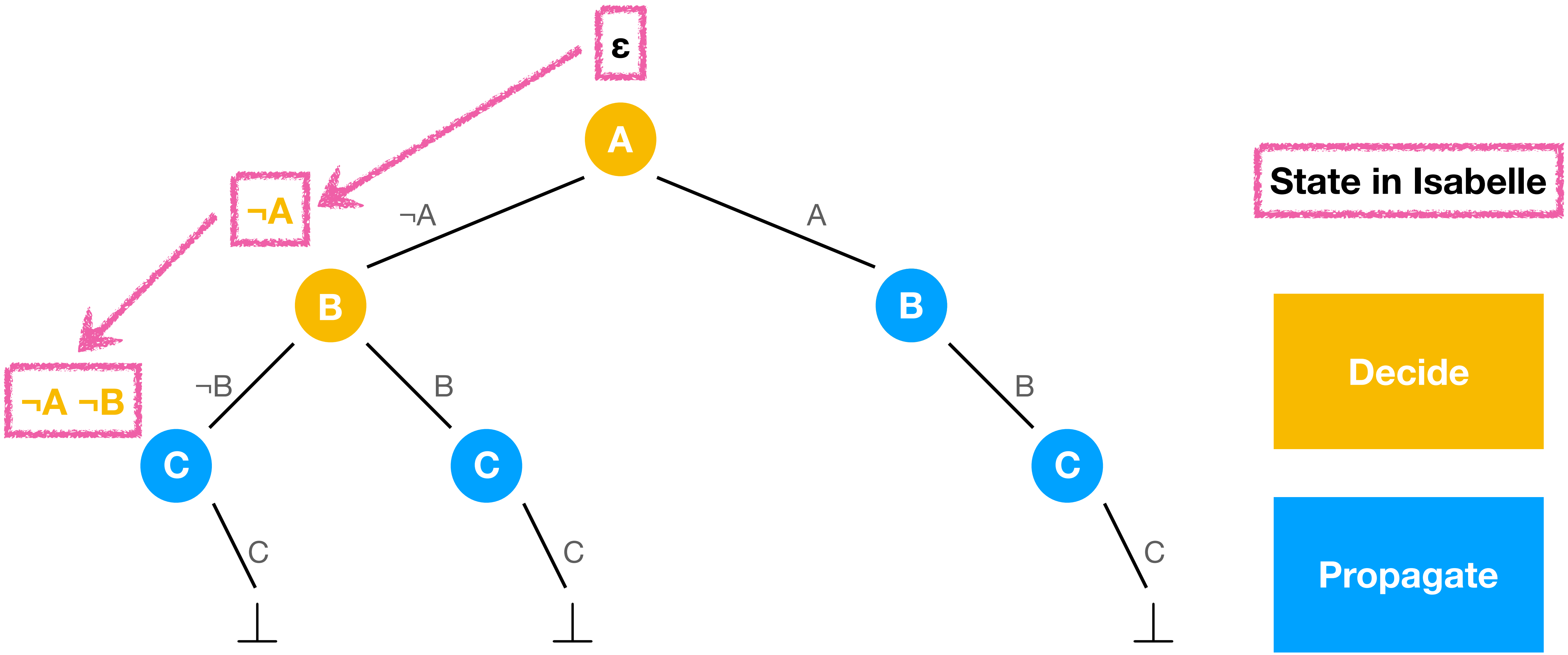
Propagate



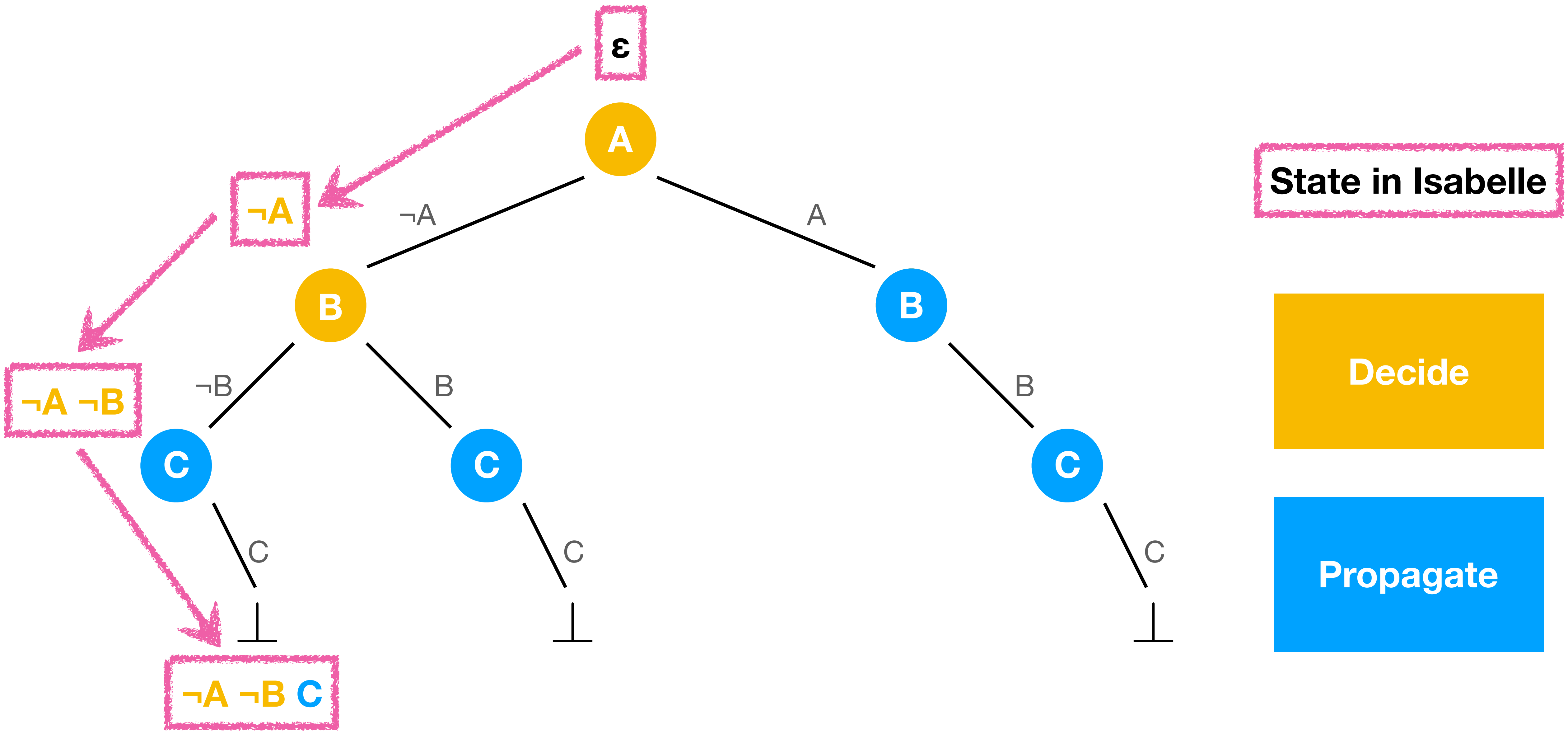
State in Isabelle

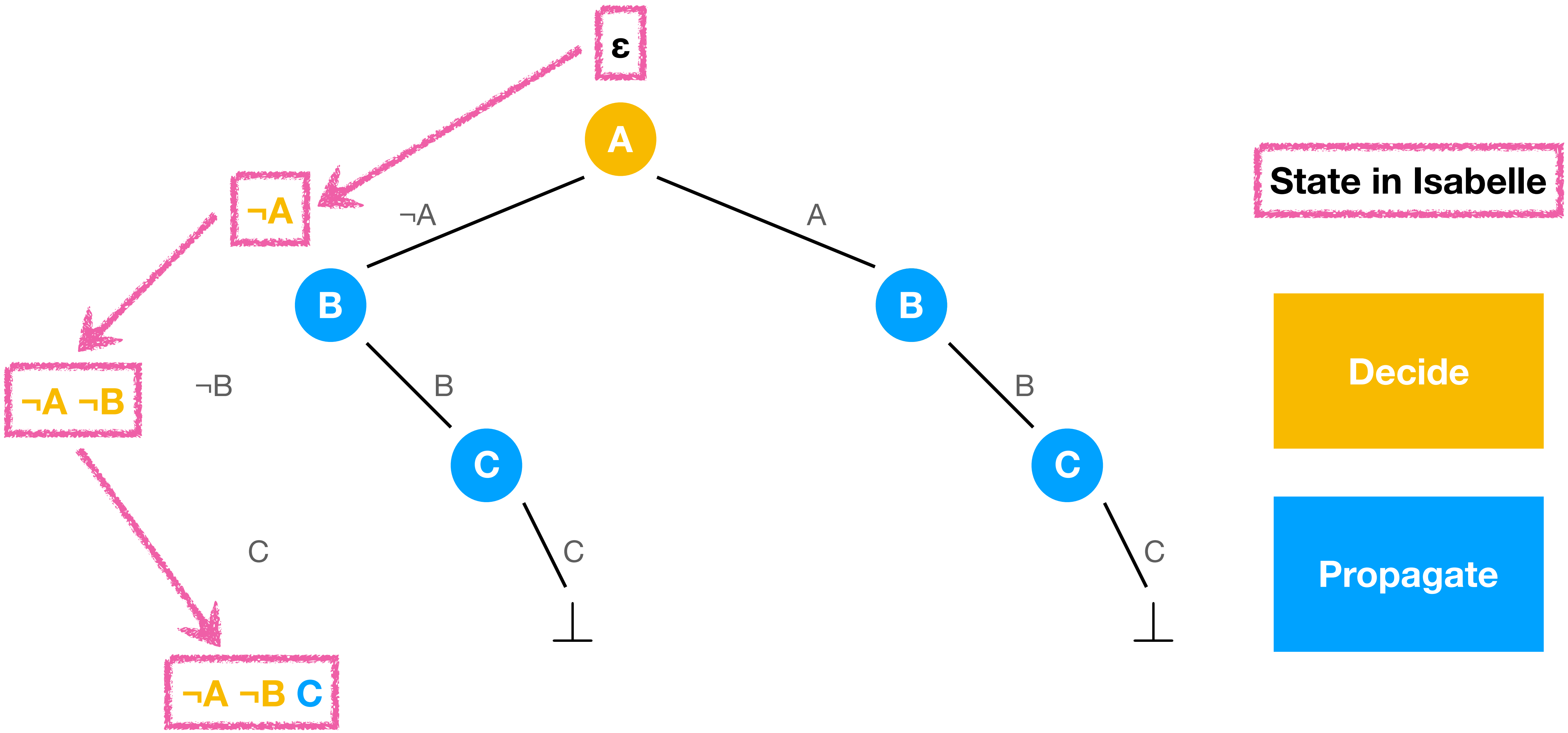
Decide

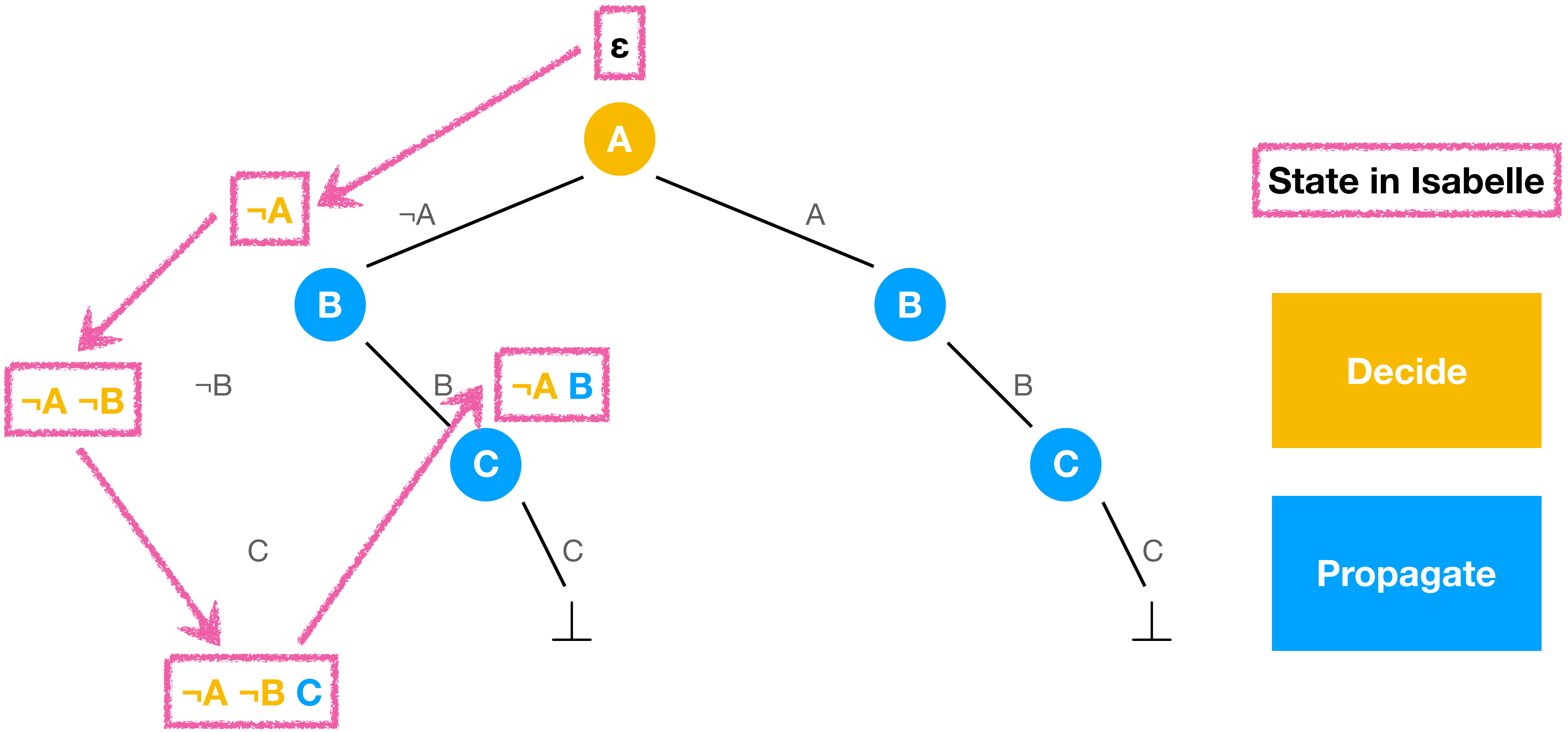
Propagate

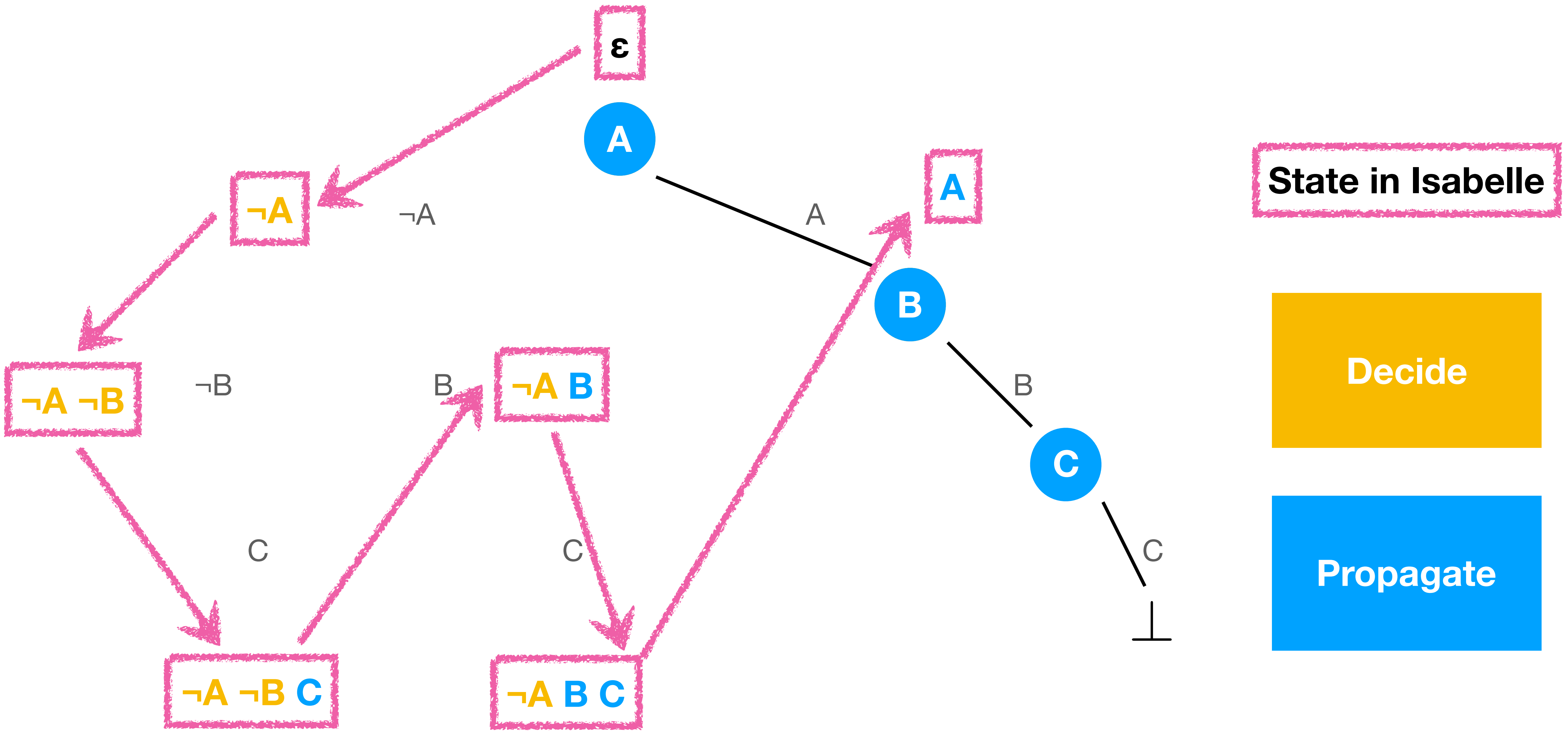


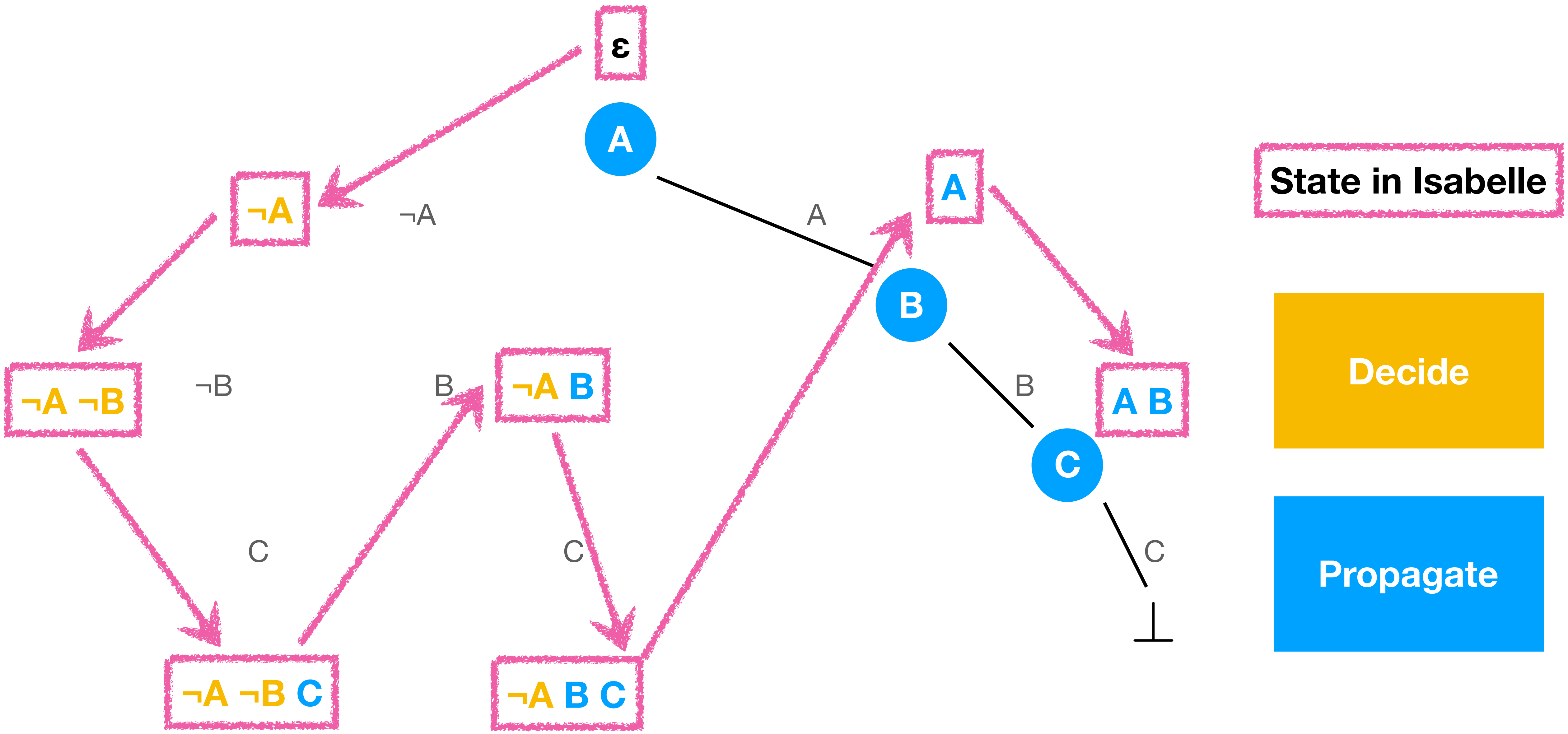
ε
A
¬A
¬A
A
B
B
¬A ¬B
¬B
B
B
C
C
C
C
C
C
⊥
⊥
⊥
State in Isabelle
Decide
Propagate

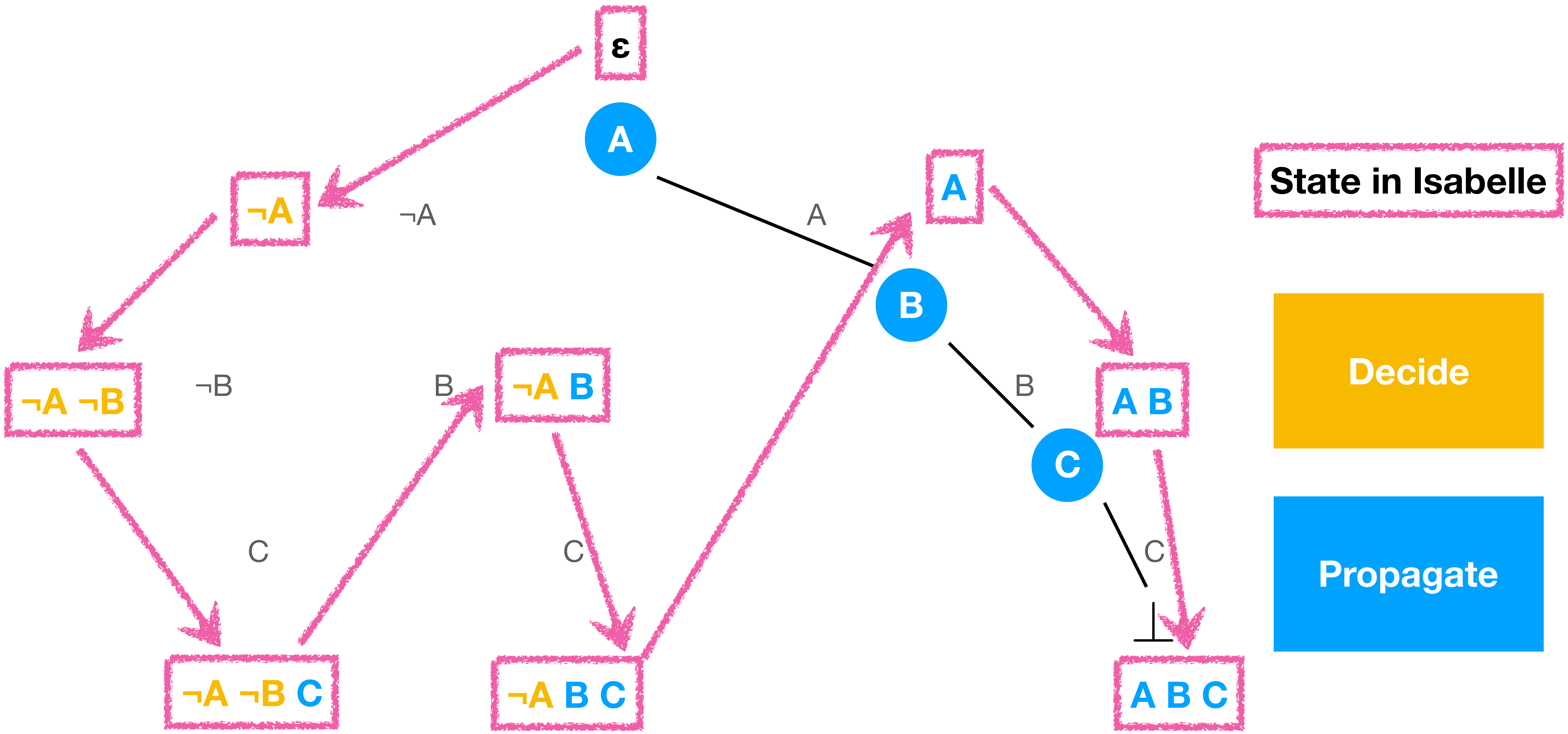

ε
A
¬A
A
¬A
B
B
¬A ¬B
¬B
B
B
C
C
C
C
C
C
⊥
⊥
⊥
¬A ¬B C
State in Isabelle
Decide
Propagate


ε
A
¬A
A
¬A
B
B
¬A ¬B
¬B
B
B
C
C
C
C
C
¬A ¬B C
⊥
⊥
State in Isabelle
Decide
Propagate


ε
A
¬A
A
¬A
B
B
¬A ¬B
¬B
B
¬A B
B
C
C
C
C
C
⊥
⊥
¬A ¬B C
State in Isabelle
Decide
Propagate


ε
A
¬A
¬A
A
A
B
¬A ¬B
¬B
B
¬A B
B
C
C
C
C
¬A ¬B C
¬A B C
⊥
State in Isabelle
Decide
Propagate


ε
A
¬A
¬A
A
A
B
¬A ¬B
¬B
B
¬A B
B
A B
C
C
C
C
¬A ¬B C
¬A B C
⊥
State in Isabelle
Decide
Propagate



No more transitions and conflict:
UNSAT

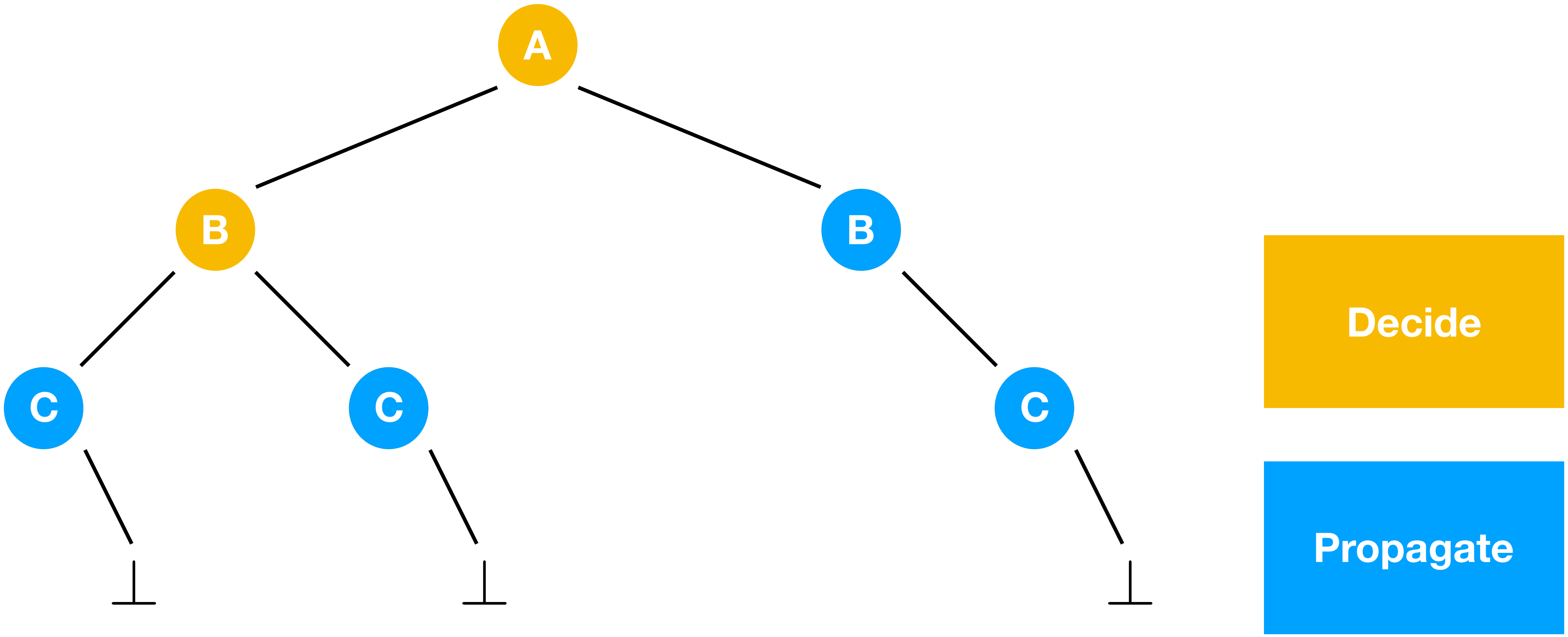
In Isabelle

State in Isabelle

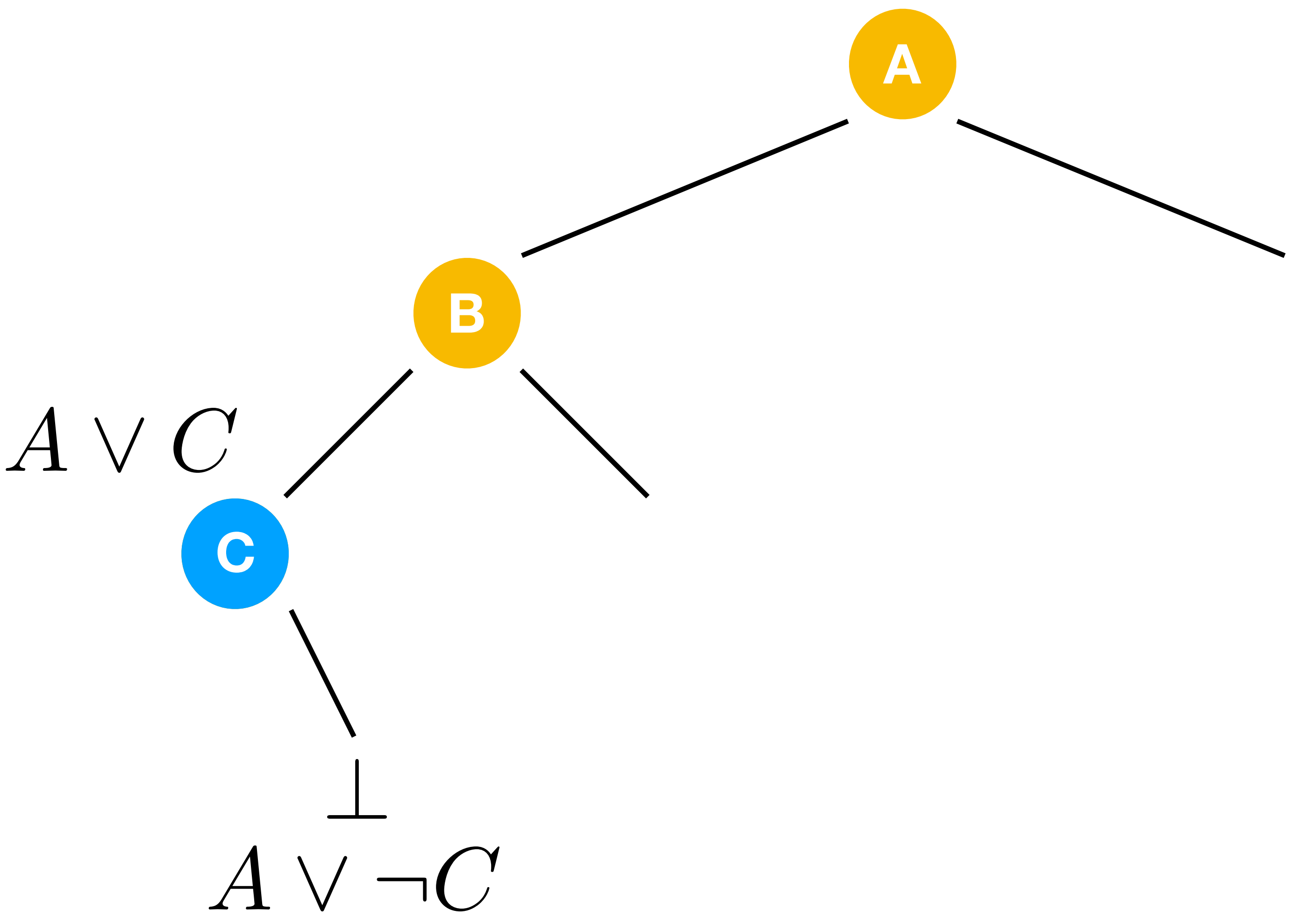
Pair path-clauses: (M, N)

Decide in Isabelle

$$\text{undefined\_lit}\ M\ L \implies L \in N \implies (M, N) \Rightarrow_{\text{CDCL}} (M\,L, N)$$


A
B
B
C
C
C
⊥
⊥
⊥
Decide
Propagate

DPLL+BJ

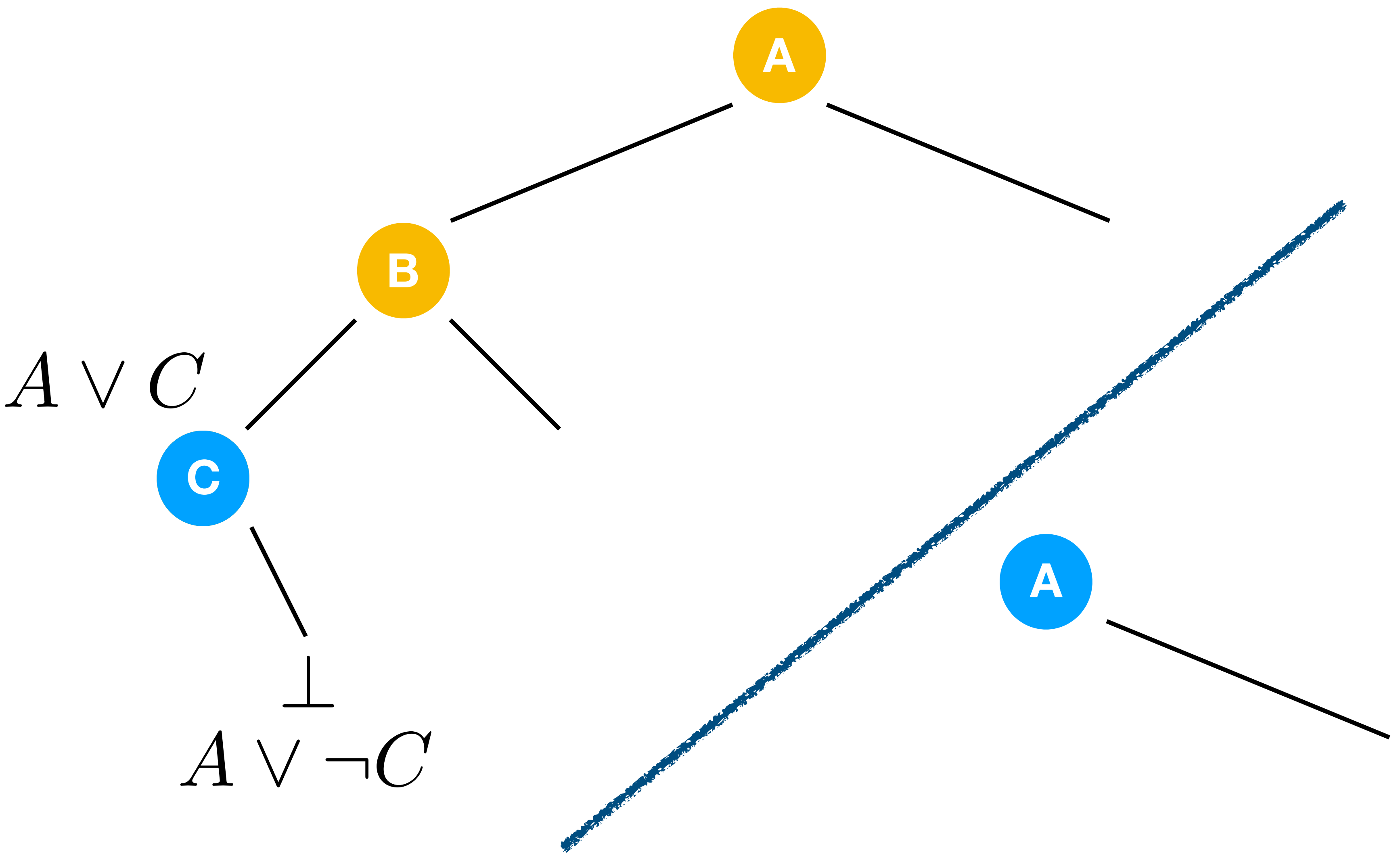


Decide

Propagate

Analyse + Backjump

DPLL+BJ

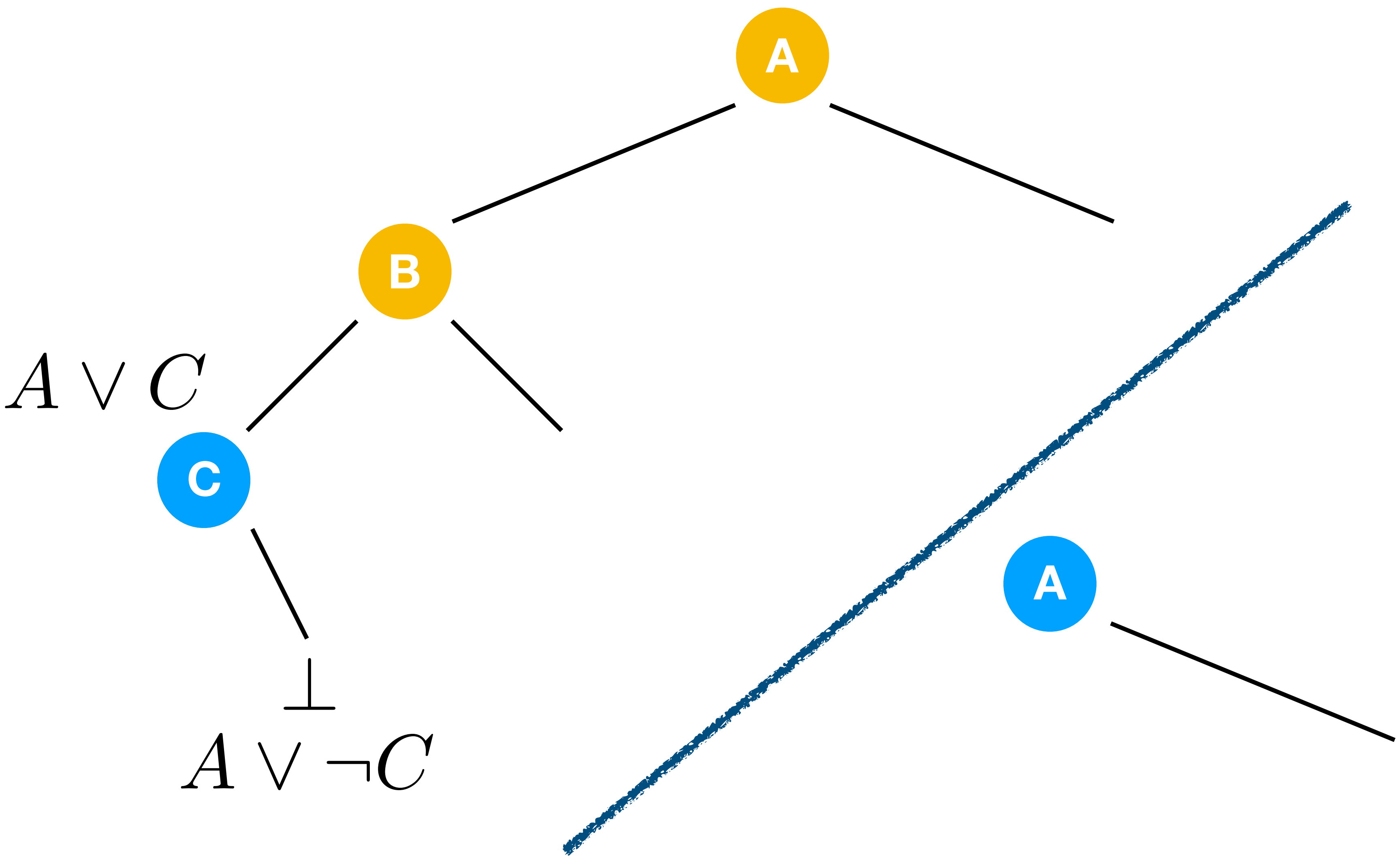


Decide

Propagate

Analyse + Backjump

CDCL



Decide

Propagate

Analyse + Backjump

Learn + forget clause

New learned clause: A

Abstract CDCL

Nieuwenhuis, Oliveras, and Tinelli 2006

↑ refines

Concrete CDCL

Weidenbach, 2015

↑ refines

CDCL with efficient data structure

Eén and Sörensson, 2004

↑ refines

Executable SAT solver

To appear

INVENTORS FOR THE DIGITAL WORLD

VRIJE UNIVERSITEIT AMSTERDAM

max planck institut informatik

Abstract CDCL

Nieuwenhuis, Oliveras, and Tinelli 2006

DPLL	DPLL+BJ	CDCL
Decide	Decide	Decide
Propagate	Propagate	Propagate
Backtrack	Analyse + Backjump	Analyse + Backjump
		Learn + forget clause

DPLL $\xrightarrow{\text{specialises}}$	DPLL+BJ	CDCL
Decide	**Decide**	**Decide**
Propagate	**Propagate**	**Propagate**
Backtrack	**Analyse + Backjump**	**Analyse + Backjump**
		Learn + forget clause

DPLL → DPLL+BJ CDCL

specialises

DPLL	DPLL+BJ	CDCL
Decide	**Decide**	**Decide**
Propagate	**Propagate**	**Propagate**
Backtrack	**Analyse + Backjump**	**Analyse + Backjump**
		Learn + forget clause

parametrise by `BJ_cond`

in Isabelle

```
submodule DPLL ⊆ DPLL+BJ where
  BJ_cond = BT_cond
```

in Isabelle

DPLL → DPLL+BJ CDCL

specialises

DPLL	DPLL+BJ	CDCL
Decide	**Decide**	**Decide**
Propagate	**Propagate**	**Propagate**
Backtrack	**Analyse + Backjump**	**Analyse + Backjump**
		Learn + forget clause

parametrise by `BJ_cond`

in Isabelle

INVENTORS FOR THE DIGITAL WORLD

VRIJE UNIVERSITEIT AMSTERDAM

max planck institut informatik

```
submodule DPLL ⊆ DPLL+BJ where
  BJ_cond = BT_cond
```

in Isabelle

DPLL → DPLL+BJ CDCL

discharge those assumptions

DPLL	DPLL+BJ	CDCL
Decide	**Decide**	**Decide**
Propagate	**Propagate**	**Propagate**
Backtrack	**Analyse + Backjump**	**Analyse + Backjump**
		Learn + forget clause

parametrise by `BJ_cond`

in Isabelle

```
submodule DPLL ⊆ DPLL+BJ where
  BJ_cond = BT_cond
```

in Isabelle

DPLL	DPLL+BJ	CDCL
Decide	**Decide**	**Decide**
Propagate	**Propagate**	**Propagate**
Backtrack	**Analyse + Backjump**	**Analyse + Backjump**
		Learn + forget clause

DPLL → DPLL+BJ: specialises

parametrise by `BJ_cond`

in Isabelle

DPLL $\xrightarrow{\text{specialises}}$	DPLL+BJ	CDCL
Decide	**Decide**	**Decide**
Propagate	**Propagate**	**Propagate**
Backtrack	**Analyse + Backjump**	**Analyse + Backjump**
		Learn + forget clause

DPLL $\xrightarrow{\text{specialises}}$ DPLL+BJ $\xleftarrow{\text{extends}}$ CDCL

DPLL	DPLL+BJ	CDCL
Decide	Decide	Decide
Propagate	Propagate	Propagate
Backtrack	Analyse + Backjump	Analyse + Backjump
		Learn + forget clause

```
CDCL = DPLL+BJ + Learn
     + Forget
```

in Isabelle

DPLL $\xrightarrow{\text{specialises}}$ DPLL+BJ $\xleftarrow{\text{extends}}$ CDCL

DPLL	DPLL+BJ	CDCL
Decide	**Decide**	**Decide**
Propagate	**Propagate**	**Propagate**
Backtrack	**Analyse + Backjump**	**Analyse + Backjump**
		Learn + forget clause

INVENTORS FOR THE DIGITAL WORLD

VRIJE UNIVERSITEIT AMSTERDAM

max planck institut informatik

DPLL	DPLL+BJ	CDCL
Decide	Decide	Decide
Propagate	Propagate	Propagate
Backtrack	Analyse + Backjump	Analyse + Backjump
		Learn + forget clause

DPLL → DPLL+BJ: specialises

CDCL → DPLL+BJ: extends

DPLL $\xrightarrow{\text{specialises}}$ DPLL+BJ $\xleftarrow{\text{extends}}$ CDCL

DPLL	DPLL+BJ	CDCL
termination	**termination**	**non-termination**

DPLL → DPLL+BJ ← CDCL

specialises extends

termination **termination** **non-termination**

Learn + forget clause

infinite chain of learn and forget

DPLL $\xrightarrow{\text{specialises}}$ DPLL+BJ $\xleftarrow{\text{extends}}$ CDCL

termination	**termination**	**non-termination**

Analyse + Backjump **Learn + forget clause**

infinite chain of learn and forget

Abstract CDCL

Nieuwenhuis, Oliveras, and Tinelli 2006

↑ refines

Concrete CDCL

Weidenbach, 2015

↑ refines

CDCL with efficient data structure

Eén and Sörensson, 2004

↑ refines

Executable SAT solver

To appear

DPLL DPLL+BJ CDCL

Abstract CDCL

Nieuwenhuis, Oliveras, and Tinelli 2006

refines

Concrete CDCL

Weidenbach, 2015

refines

CDCL with efficient data structure

Eén and Sörensson, 2004

refines

Executable SAT solver

To appear

Concrete CDCL

Weidenbach, 2015

Backjump

on paper

$$\begin{array}{l}\textbf{if } \mathbf{C} \in \mathbf{N} \textbf{ and } \mathbf{M} \models \neg\mathbf{C} \\ \textbf{and there is } \mathbf{C}' \textbf{ such that } \ldots \\ (\mathbf{M}, \mathbf{N}) \Rightarrow (\mathbf{L}\,\mathbf{M}', \mathbf{N})\end{array}$$

How do we get a suitable C'?

INVENTORS FOR THE DIGITAL WORLD

VRIJE UNIVERSITEIT AMSTERDAM

max planck institut informatik

Backjump

on paper

$\mathbf{if\ C \in N\ and\ M \vDash \neg C}$

$\mathbf{and\ there\ is\ C'\ such\ that\ \ldots}$

$\mathbf{(M, N) \Rightarrow (L\, M', N)}$

How do we get a suitable C'?

- First unique implication point

Theorem (no relearning):
No clause can be learned twice.

Theorem (no relearning):
No clause can be learned twice.

Proof. By contradiction. Assume CDCL learns the same clause twice, i.e., it reaches a state $(M;N;U;k;D \vee L)$ where Backtracking is applicable and $D \vee L \in (N \cup U)$.

More precisely, the state has the form $(M_1K_i^{i+1}M_2K_1^kK_2 \ldots K_n;N;U;k;D \vee L)$ where the K_i, $i > 1$ are propagated literals that do not occur complemented in D, as for otherwise D cannot be of level i. Furthermore, one of the K_i is the complement of L.

But now, because $D \vee L$ is false in $M_1K_i^{i+1}M_2K_1^kK_2 \ldots K_n$ and $D \vee L \in (N \cup U)$

instead of deciding K_1^k the literal L should be propagated by a reasonable strategy. A contradiction. Note that none of the K_i can be annotated with $D \vee L$.

‹700 lines of proof› in Isabelle

Theorem (no relearning):
No clause can be learned twice.

Proof. By contradiction. Assume CDCL learns the same clause twice, i.e., it reaches a state $(M;N;U;k;D \lor L)$ where Backtracking is applicable and $D \lor L \in (N \cup U)$.

More precisely, the state has the form $(M_1K_i^{i+1}M_2K_1^kK_2 \ldots K_n;N;U;k;D \lor L)$ where the K_i, $i > 1$ are propagated literals that do not occur complemented in D, as for otherwise D cannot be of level i. Furthermore, one of the K_i is the complement of L.

But now, because $D \lor L$ is false in $M_1K_i^{i+1}M_2K_1^kK_2 \ldots K_n$ and $D \lor L \in (N \cup U)$

instead of deciding K_1^k the literal L should be propagated by a reasonable strategy. A contradiction. Note that none of the K_i can be annotated with $D \lor L$.

INVENTORS FOR THE DIGITAL WORLD

VRIJE UNIVERSITEIT AMSTERDAM

max planck institut informatik

Theorem (no relearning):

No clause can be learned twice.

Proof. By contradiction. Assume CDCL learns the same clause twice, i.e., it reaches a state $(M;N;U;k;D \lor L)$ where Backtracking is applicable and $D \lor L \in (N \cup U)$.

More precisely, the state has the form $(M_1K^{i+1}M_2K_1{}^{k}K_2 \ldots K_n;N;U;k;D\lor L)$ where the K_i, $i > 1$ are propagated literals that do not occur complemented in D, as for otherwise D cannot be of level i. Furthermore, one of the K_i is the complement of L.

But now, because $D \lor L$ is false in $M_1K^{i+1}M_2K_1{}^{k}K_2 \ldots K_n$ and $D \lor L \in (N \cup U)$

instead of deciding $K_1{}^{k}$ the literal L should be propagated by a reasonable strategy. A contradiction. Note that none of the K_i can be annotated with $D \lor L$.

$\boxed{\varepsilon} \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \ldots \rightarrow \rightarrow \rightarrow$

Theorem (no relearning):

No clause can be learned twice.

Proof. By contradiction. Assume CDCL learns the same clause twice, i.e., it reaches a state $(M;N;U;k;D \vee L)$ where Backtracking is applicable and $D \vee L \in (N \cup U)$.

More precisely, the state has the form $(M_1K^{i+1}M_2K_1{}^kK_2 \ldots K_n;N;U;k;D \vee L)$ where the K_i, $i > 1$ are propagated literals that do not occur complemented in D, as for otherwise D cannot be of level i. Furthermore, one of the K_i is the complement of L.

But now, because $D \vee L$ is false in $M_1K^{i+1}M_2K_1{}^kK_2 \ldots K_n$ and $D \vee L \in (N \cup U)$

instead of deciding $K_1{}^k$ the literal L should be propagated by a reasonable strategy. A contradiction. Note that none of the K_i can be annotated with $D \vee L$.

Theorem (no relearning):
No clause can be learned twice.

Proof. By contradiction. Assume CDCL learns the same clause twice, i.e., it reaches a state $(M;N;U;k;D \vee L)$ where Backtracking is applicable and $D \vee L \in (N \cup U)$.

More precisely, the state has the form $(M_1K^{i+1}M_2K_1{}^kK_2 \ldots K_n;N;U;k;D \vee L)$ where the K_i, $i > 1$ are propagated literals that do not occur complemented in D, as for otherwise D cannot be of level i. Furthermore, one of the K_i is the complement of L.

But now, because $D \vee L$ is false in $M_1K^{i+1}M_2K_1{}^kK_2 \ldots K_n$ and $D \vee L \in (N \cup U)$

instead of deciding $K_1{}^k$ the literal L should be propagated by a reasonable strategy. A contradiction. Note that none of the K_i can be annotated with $D \vee L$.

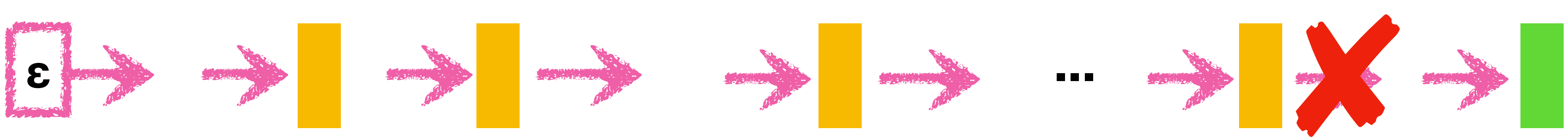
Theorem (no relearning):
No clause can be learned twice.

Proof. By contradiction. Assume CDCL learns the same clause twice, i.e., it reaches a state $(M;N;U;k;D \vee L)$ where Backtracking is applicable and $D \vee L \in (N \cup U)$.

More precisely, the state has the form $(M_1K^{i+1}M_2K_1^kK_2 \ldots K_n;N;U;k;D\vee L)$ where the K_i, $i > 1$ are propagated literals that do not occur complemented in D, as for otherwise D cannot be of level i. Furthermore, one of the K_i is the complement of L.

But now, because $D \vee L$ is false in $M_1K^{i+1}M_2K_1^kK_2 \ldots K_n$ and $D \vee L \in (N \cup U)$

instead of deciding K_1^k the literal L should be propagated by a reasonable strategy. A contradiction. Note that none of the K_i can be annotated with $D \vee L$.



Abstract CDCL

Nieuwenhuis, Oliveras, and Tinelli 2006

↑ refines

Concrete CDCL

Weidenbach, 2015

↑ refines

CDCL with efficient datastrucure

Eén and Sörensson, 2004

↑ refines

Executable SAT solver

To appear

CDCL with efficient datastructure

- Two watched literals: important for performance
- Nice to have formally

How hard is it?

	Paper	Proof assistant
Abstract CDCL	13 pages	50 pages
Concrete CDCL	9 pages (½ month)	90 pages (5 months)
Two-Watched	1 page (C++ code of MiniSat)	265 pages (9 months)

Inria INVENTORS FOR THE DIGITAL WORLD VU VRIJE UNIVERSITEIT AMSTERDAM max planck institut informatik

Conclusion

Concrete outcome

- verified SAT solver framework
- verified executable SAT solver
- improve book draft

Methodology

- Refinement

Future work

- SAT Modulo Theories (e.g., CVC or z3)

INVENTORS FOR THE DIGITAL WORLD

VRIJE UNIVERSITEIT AMSTERDAM

max planck institut informatik