A First-Order Davis-Putnam-Logemann-Loveland Procedure

Peter Baumgartner
Institut für Informatik
Universität Koblenz-Landau
Germany

+ Some slides from "First Order Theorem Proving" Tutorial

+ Some slides from "Model Evolution Calculus" Talk

+ Some editing and extra slides from me :)

CSC2542 Automated Theorem Proving
Stavros Vassos

http://www.uni-koblenz.de/~peter/

Two Separated Worlds

	First-Order Reasoning	Propositional Reasoning
Techniques	Resolution	DPLL
	Model Elimination	OBDD
	Hyper Linking	Stalmarck's Method
		Tableaux
		Stochastic (GSAT)
Systems	E, Otter, Setheo, SNARK, Spass, Vampire	Chaff, SMV, Heerhugo, FACT, WalkSat
Applications	SW-Verification (Limited)	Symbolic Model Checking
	Mathematics	Mathematics
	Discourse Representation	Planning, Description Logics
	TPTP	Nonmonotonic Reasoning

Can couple these worlds more closely?

Overview

- Intro to First-Order Theorem Proving
- FDPLL Motivation
- DPLL as a Semantic Tree Method
- FDPLL as a First-Order Semantic Tree Method
- Soundness and Completeness of FDPLL
- Discussion

Refutational Theorem Proving

- Suppose we want to prove $H \models G$.
- Equivalently, we can prove that $F := H \rightarrow G$ is valid.
- Equivalently, we can prove that $\neg F$, i.e.$H \wedge \neg G$ is unsatisfiable.

This principle of “refutational theorem proving” is the basis of almost all automated theorem proving methods.

Normal Forms

Study of normal forms motivated by

- reduction of logical concepts,
- efficient data structures for theorem proving.

The main problem in first-order logic is the treatment of quantifiers. The subsequent normal form transformations are intended to eliminate many of them.

Prenex Normal Form

Prenex formulas have the form

$$Q_1 x_1 \ldots Q_n x_n \; F,$$

where F is quantifier-free and $Q_i \in \{\forall, \exists\}$;
we call $Q_1 x_1 \ldots Q_n x_n$ the **quantifier prefix** and F the **matrix** of the formula.

Skolemization

Intuition: replacement of $\exists y$ by a concrete choice function computing y from all the arguments y depends on.

$$\forall x_1, \ldots, x_n \exists y F \quad \Rightarrow_S \quad \forall x_1, \ldots, x_n F[f(x_1, \ldots, x_n)/y]$$

where f/n is a new function symbol (**Skolem function**).

Skolemization

Together: $F \overset{*}{\Rightarrow}_P \underbrace{G}_{\text{prenex}} \overset{*}{\Rightarrow}_S \underbrace{H}_{\text{prenex, no } \exists}$

Theorem: The given and the final formula are equi-satisfiable.

The Complete Picture

$$
\begin{array}{llll}
F & \overset{*}{\Rightarrow}_P & Q_1 y_1 \ldots Q_n y_n \; G & (G \text{ quantifier-free}) \\
 & \overset{*}{\Rightarrow}_S & \forall x_1, \ldots, x_m \; H & (m \leq n,\ H \text{ quantifier-free}) \\
 & \overset{*}{\Rightarrow}_K & \underbrace{\underbrace{\forall x_1, \ldots, x_m}_{\text{leave out}} \; \bigwedge_{i=1}^{k} \underbrace{\bigvee_{j=1}^{n_i} L_{ij}}_{\text{clauses } C_i}}_{F'} &
\end{array}
$$

$N = \{C_1, \ldots, C_k\}$ is called the **clausal (normal) form** (CNF) of F.

Note: the variables in the clauses are implicitly universally quantified.

The Complete Picture

$$
\begin{aligned}
F \quad & \overset{*}{\Rightarrow}_P \quad Q_1 y_1 \ldots Q_n y_n \; G && (G \text{ quantifier-free}) \\
& \overset{*}{\Rightarrow}_S \quad \forall x_1, \ldots, x_m \; H && (m \leq n,\, H \text{ quantifier-free}) \\
& \overset{*}{\Rightarrow}_K \quad \underbrace{\underbrace{\forall x_1, \ldots, x_m}_{\text{leave out}} \; \bigwedge_{i=1}^{k} \underbrace{\bigvee_{j=1}^{n_i} L_{ij}}_{\text{clauses } C_i}}_{F'}
\end{aligned}
$$

$N = \{C_1, \ldots, C_k\}$ is called the **clausal (normal) form** (CNF) of F.

Note: the variables in the clauses are implicitly universally quantified.

Now we arrived at "low-level predicate logic" and the proof problem, proper, i.e. to prove that the clause set is unsatisfiable.

Herbrand Theory

Some thoughts

- Suppose we want to prove $H \models G$.
- Equivalently, we can prove that $F := H \land \neg G$ is unsatisfiable.
- We have seen how F can be syntactically simplified to clause form F' in a satisfiability preserving way.
- It remains to prove that F' is unsatisfiable.

Herbrand Theory

Some thoughts

- Suppose we want to prove $H \models G$.
- Equivalently, we can prove that $F := H \land \neg G$ is unsatisfiable.
- We have seen how F can be syntactically simplified to clause form F' in a satisfiability preserving way.
- It remains to prove that F' is unsatisfiable.
- Does this mean that "**all** interpretations have to be searched"?
  No! It suffices to "search only through Herbrand interpretations"

Herbrand Interpretations

values are fixed to be ground terms and **functions are fixed** to be the **term constructors**. Only predicate symbols may be freely interpreted as relations .

Proposition
Every set of ground atoms I uniquely determines a Herbrand interpretation $\mathcal{A}$ via

$$(s_1, \ldots, s_n) \in p_{\mathcal{A}} \quad :\Leftrightarrow \quad p(s_1, \ldots, s_n) \in I$$

Thus we shall identify Herbrand interpretations (over Σ) with sets of Σ-ground atoms.

FDPLL Motivation

- Propositional Case:
  - SAT: Is the set of propositional clauses C unsatisfiable?
  - View **DPLL** as a way to **search** for a model among all **propositional Herbrand interpretations**.
- First-Order Case:
  - Refutational theorem proving: Is the set of FOL clauses C unsatisfiable?
  - **FDPLL**: Use the same ideas as in DPLL to **search** for a model among all **FOL Herbrand interpretations**
  - Take advantage of efficient methods for unification, subsumption, …

Propositional DPLL as a Semantic Tree Method

(1) $\mathbf{A} \lor \mathbf{B}$ (2) $\mathbf{C} \lor \neg\mathbf{A}$ (3) $\mathbf{D} \lor \neg\mathbf{C} \lor \neg\mathbf{A}$ (4) $\neg\mathbf{D} \lor \neg\mathbf{B}$

⟨empty tree⟩

$\{\} \not\models \mathbf{A} \lor \mathbf{B}$

$\{\} \models \mathbf{C} \lor \neg\mathbf{A}$

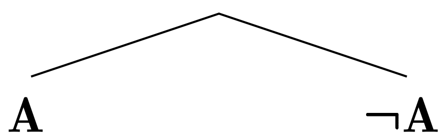
$\{\} \models \mathbf{D} \lor \neg\mathbf{C} \lor \neg\mathbf{A}$

$\{\} \models \neg\mathbf{D} \lor \neg\mathbf{B}$

- ✗ A Branch stands for an interpretation
- ✗ Purpose of splitting: Satisfy a clause that is currently “false”
- ✗ Close branch if some clause plainly contradicts it ($\star$)

Propositional DPLL as a Semantic Tree Method

(1) $\mathbf{A} \vee \mathbf{B}$ (2) $\mathbf{C} \vee \neg\mathbf{A}$ (3) $\mathbf{D} \vee \neg\mathbf{C} \vee \neg\mathbf{A}$ (4) $\neg\mathbf{D} \vee \neg\mathbf{B}$

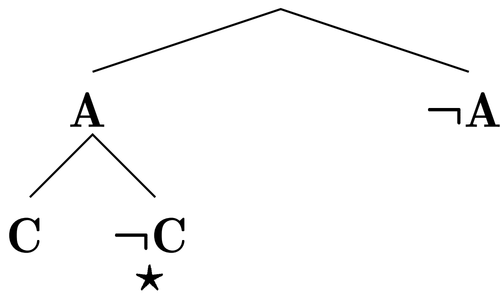


$$\{\mathbf{A}\} \models \mathbf{A} \vee \mathbf{B}$$
$$\{\mathbf{A}\} \not\models \mathbf{C} \vee \neg\mathbf{A}$$
$$\{\mathbf{A}\} \models \mathbf{D} \vee \neg\mathbf{C} \vee \neg\mathbf{A}$$
$$\{\mathbf{A}\} \models \neg\mathbf{D} \vee \neg\mathbf{B}$$

- ✗ A Branch stands for an interpretation
- ✗ Purpose of splitting: Satisfy a clause that is currently “false”
- ✗ Close branch if some clause plainly contradicts it ($\star$)

Propositional DPLL as a Semantic Tree Method

(1) $A \vee B$ (2) $C \vee \neg A$ (3) $D \vee \neg C \vee \neg A$ (4) $\neg D \vee \neg B$

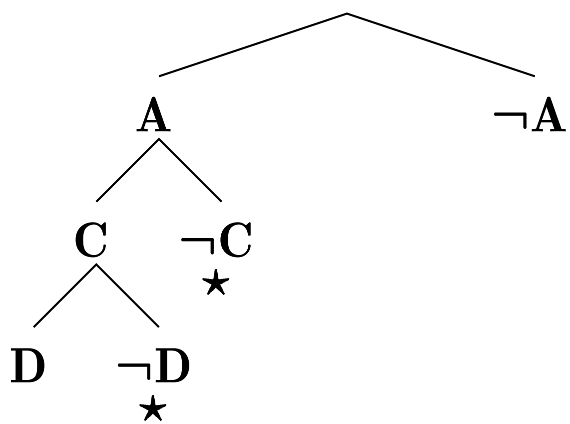


$$\{A, C\} \models A \vee B$$
$$\{A, C\} \models C \vee \neg A$$
$$\{A, C\} \not\models D \vee \neg C \vee \neg A$$
$$\{A, C\} \models \neg D \vee \neg B$$

- ✗ A Branch stands for an interpretation
- ✗ Purpose of splitting: Satisfy a clause that is currently "false"
- ✗ Close branch if some clause plainly contradicts it ($\star$)

Propositional DPLL as a Semantic Tree Method

(1) $\mathbf{A} \lor \mathbf{B}$ (2) $\mathbf{C} \lor \neg \mathbf{A}$ (3) $\mathbf{D} \lor \neg \mathbf{C} \lor \neg \mathbf{A}$ (4) $\neg \mathbf{D} \lor \neg \mathbf{B}$



$\{\mathbf{A}, \mathbf{C}, \mathbf{D}\} \models \mathbf{A} \lor \mathbf{B}$

$\{\mathbf{A}, \mathbf{C}, \mathbf{D}\} \models \mathbf{C} \lor \neg \mathbf{A}$

$\{\mathbf{A}, \mathbf{C}, \mathbf{D}\} \models \mathbf{D} \lor \neg \mathbf{C} \lor \neg \mathbf{A}$

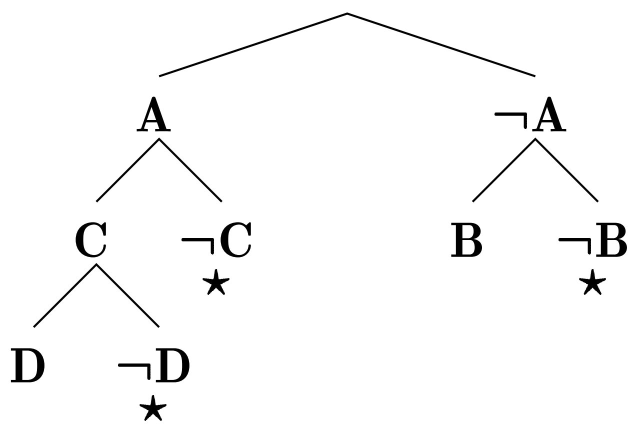
$\{\mathbf{A}, \mathbf{C}, \mathbf{D}\} \models \neg \mathbf{D} \lor \neg \mathbf{B}$

Model $\{\mathbf{A}, \mathbf{C}, \mathbf{D}\}$ found.

- ✗ A Branch stands for an interpretation
- ✗ Purpose of splitting: Satisfy a clause that is currently "false"
- ✗ Close branch if some clause plainly contradicts it ($\star$)

Propositional DPLL as a Semantic Tree Method

(1) $A \lor B$ (2) $C \lor \neg A$ (3) $D \lor \neg C \lor \neg A$ (4) $\neg D \lor \neg B$



$\{B\} \models A \lor B$

$\{B\} \models C \lor \neg A$

$\{B\} \models D \lor \neg C \lor \neg A$

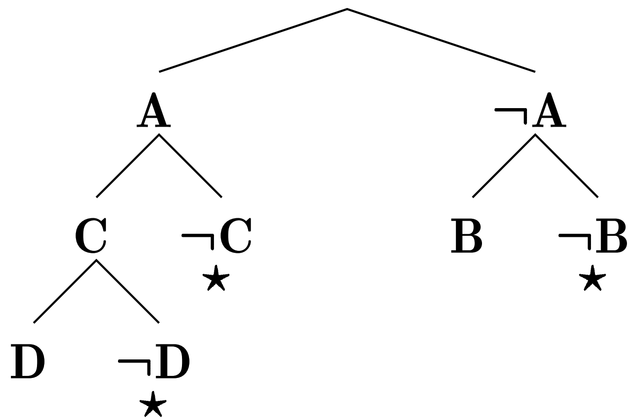
$\{B\} \models \neg D \lor \neg B$

Model $\{B\}$ found.

- ✗ A Branch stands for an interpretation
- ✗ Purpose of splitting: Satisfy a clause that is currently "false"
- ✗ Close branch if some clause plainly contradicts it ($\star$)

Propositional DPLL as a Semantic Tree Method

(1) $A \vee B$ (2) $C \vee \neg A$ (3) $D \vee \neg C \vee \neg A$ (4) $\neg D \vee \neg B$



$$\{B\} \models A \vee B$$
$$\{B\} \models C \vee \neg A$$
$$\{B\} \models D \vee \neg C \vee \neg A$$
$$\{B\} \models \neg D \vee \neg B$$

Model $\{B\}$ found.

- ✗ A Branch stands for an interpretation
- ✗ Purpose of splitting: Satisfy a clause that is currently “false”
- ✗ Close branch if some clause plainly contradicts it ($\star$)
- ✗ **Sound and complete**

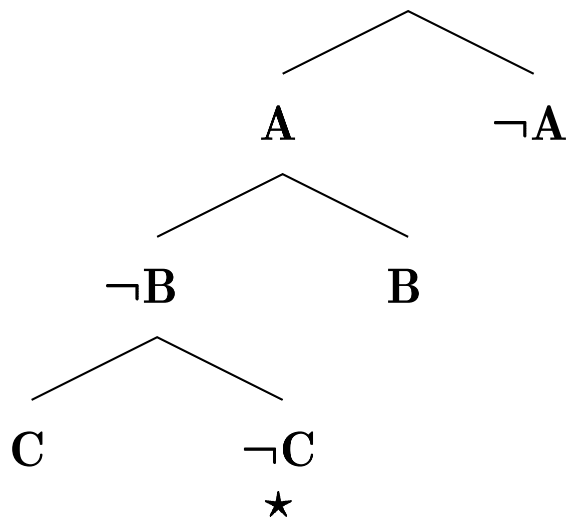
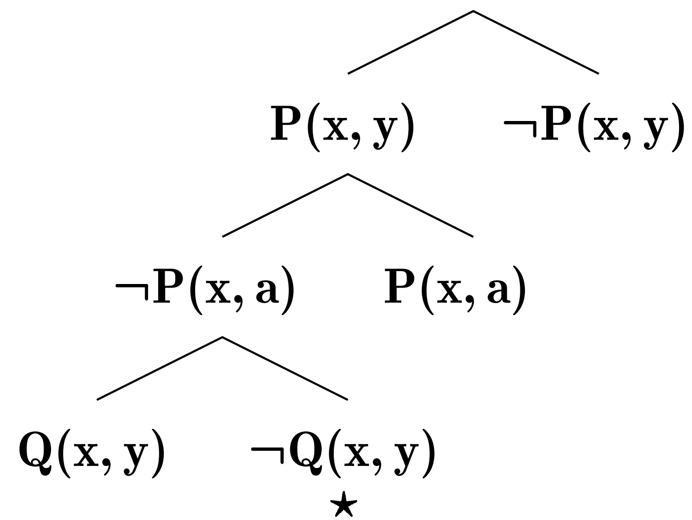
Meta-Level Strategy

Lifted data structures:

	Propositional Reasoning	First-Order Reasoning
Resolution	$A \vee \neg B \vee C$	$P(x, y) \vee \neg Q(x, z) \vee R(y, z)$

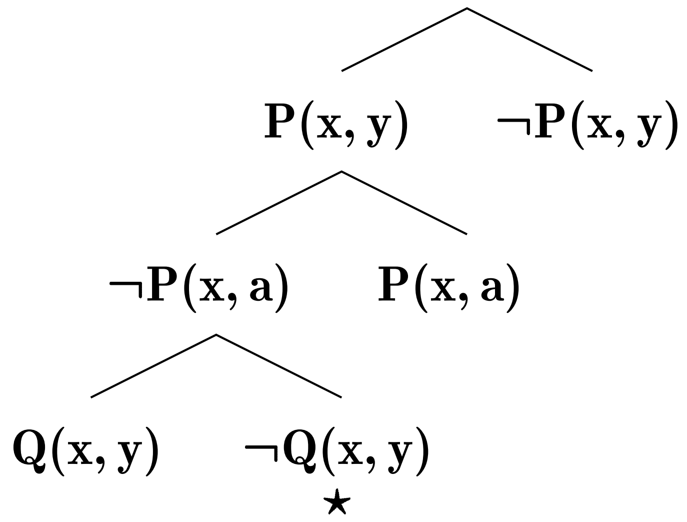
Meta-Level Strategy

Lifted data structures:

	Propositional Reasoning	First-Order Reasoning
Resolution	$A \vee \neg B \vee C$	$P(x, y) \vee \neg Q(x, z) \vee R(y, z)$
DPLL		

FDPLL: First-Order Semantic Trees

First-Order Semantic Trees

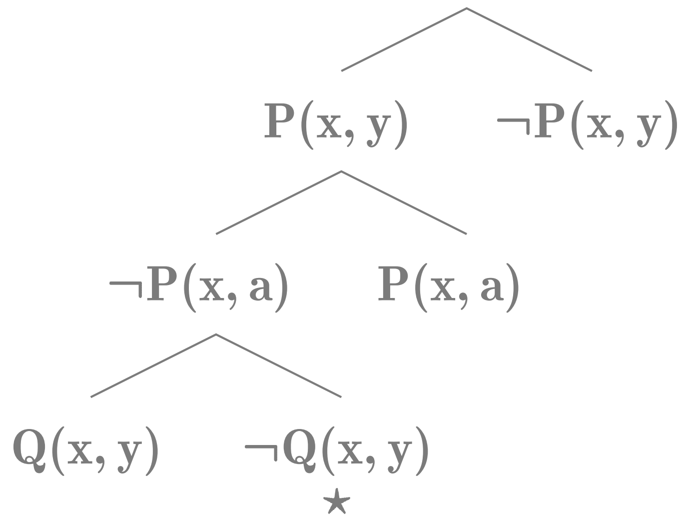


Issues:

- ✗ How are variables treated?

  (a) Universal, as in Resolution?, (b) Rigid, as in Tableaux? (c) Schema!
- ✗ How to extract an interpretation from a branch?
- ✗ When is a branch closed?
- ✗ How to construct such trees (calculus)?

First-Order Semantic Trees



Issues:

- ✗ How are variables treated?
  (a) Universal, as in Resolution?, (b) Rigid, as in Tableaux? (c) Schema!
- ✗ **How to extract an interpretation from a branch?**
- ✗ When is a branch closed?
- ✗ How to construct such trees (calculus)?

Extracting an Interpretation from a Branch

Branch $\mathcal{B}$:

$$\mathbf{P(x, y)}$$

Interpretation $[\![\mathcal{B}]\!] = \{\cdot^{\cdot^{\cdot}}\}$:

✗ A branch literal specifies the truth values for all its ground instances, unless there is a more specific literal specifying opposite truth values.

Extracting an Interpretation from a Branch

Branch $\mathcal{B}$:

Interpretation $[\![\mathcal{B}]\!] = \{\ldots\}$:

$\mathbf{P(x, y)} \longrightarrow$

$\mathbf{P(a, a)}$ $\quad$ $\mathbf{P(b, a)}$

$\mathbf{P(a, b)}$ $\quad$ $\mathbf{P(b, b)}$

✗ A branch literal specifies the truth values for all its ground instances, unless there is a more specific literal specifying opposite truth values.

Extracting an Interpretation from a Branch

Branch $\mathcal{B}$:

$P(x, y)$

$\neg P(a, y)$

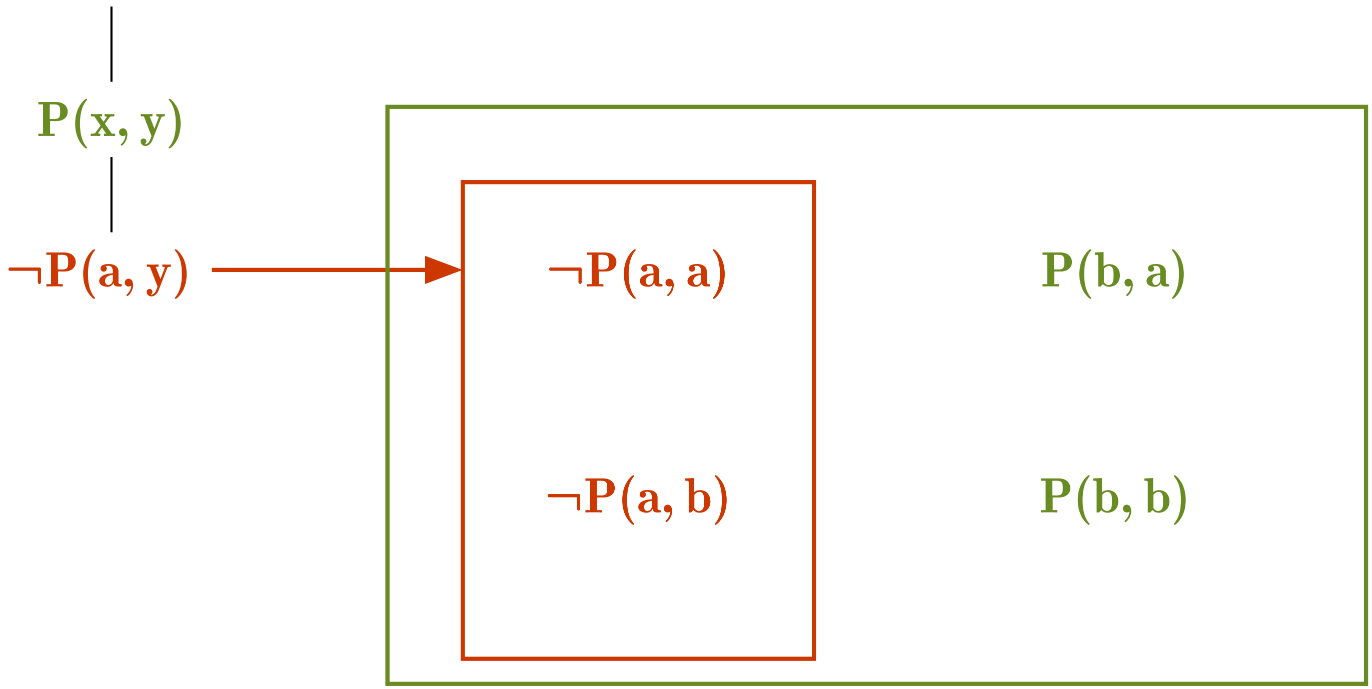
Interpretation $[\![\mathcal{B}]\!] = \{\ldots\}$:

$P(a, a)$	$P(b, a)$
$P(a, b)$	$P(b, b)$

✗ A branch literal specifies the truth values for all its ground instances, unless there is a more specific literal specifying opposite truth values.

Extracting an Interpretation from a Branch

Branch $\mathcal{B}$: Interpretation $[\![\mathcal{B}]\!] = \{\cdot^{\cdot^{\cdot}}\}$:



✗ A branch literal specifies the truth values for all its ground instances, unless there is a more specific literal specifying opposite truth values.

Extracting an Interpretation from a Branch

Branch $\mathcal{B}$:

$P(x, y)$

$\neg P(a, y)$

$\neg P(b, b)$

Interpretation $[\![\mathcal{B}]\!] = \{\cdot^{\cdot^{\cdot}}\}$:

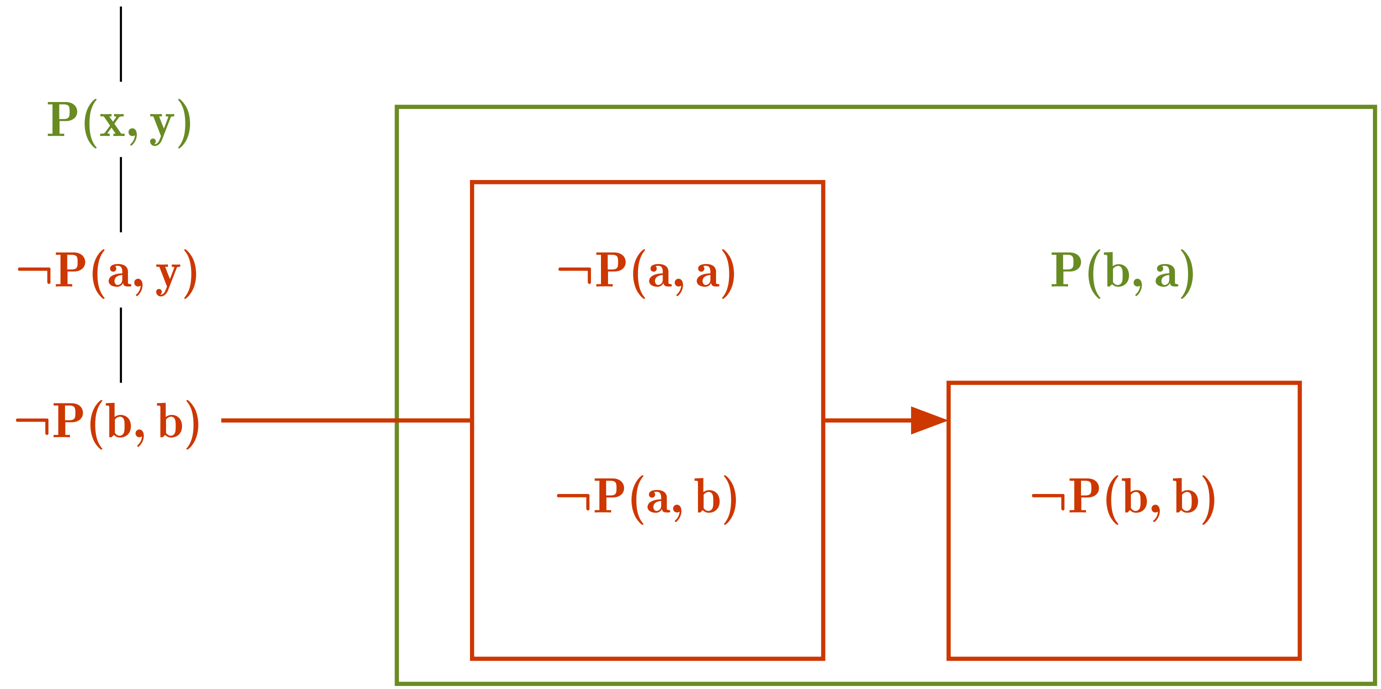
$\neg P(a, a)$ $\quad$ $P(b, a)$

$\neg P(a, b)$ $\quad$ $P(b, b)$

✗ A branch literal specifies the truth values for all its ground instances, unless there is a more specific literal specifying opposite truth values.

Extracting an Interpretation from a Branch

Branch $\mathcal{B}$: Interpretation $[\![\mathcal{B}]\!] = \{\ldots\}$:

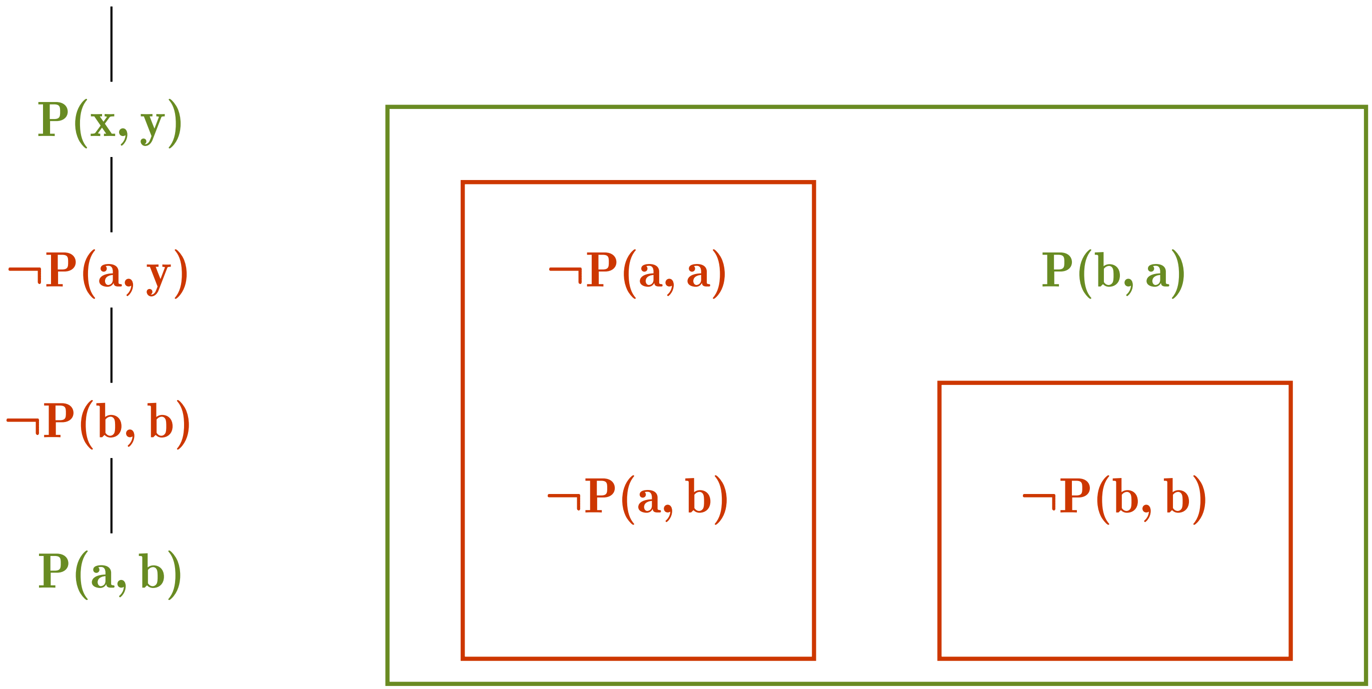


✗ A branch literal specifies the truth values for all its ground instances, unless there is a more specific literal specifying opposite truth values.

Extracting an Interpretation from a Branch

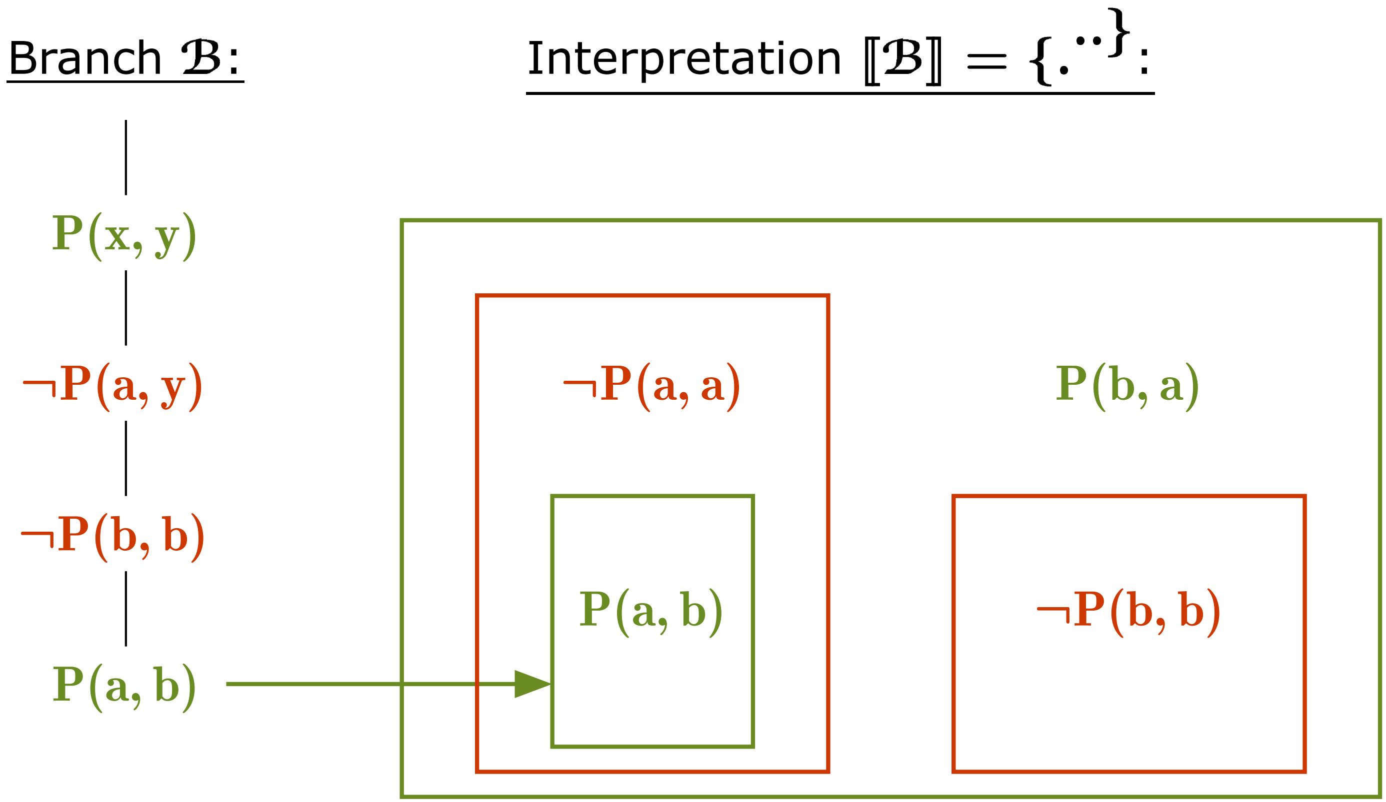
Branch $\mathcal{B}$:

Interpretation $[\![\mathcal{B}]\!] = \{\ldots\}$:



✗ A branch literal specifies the truth values for all its ground instances, unless there is a more specific literal specifying opposite truth values.

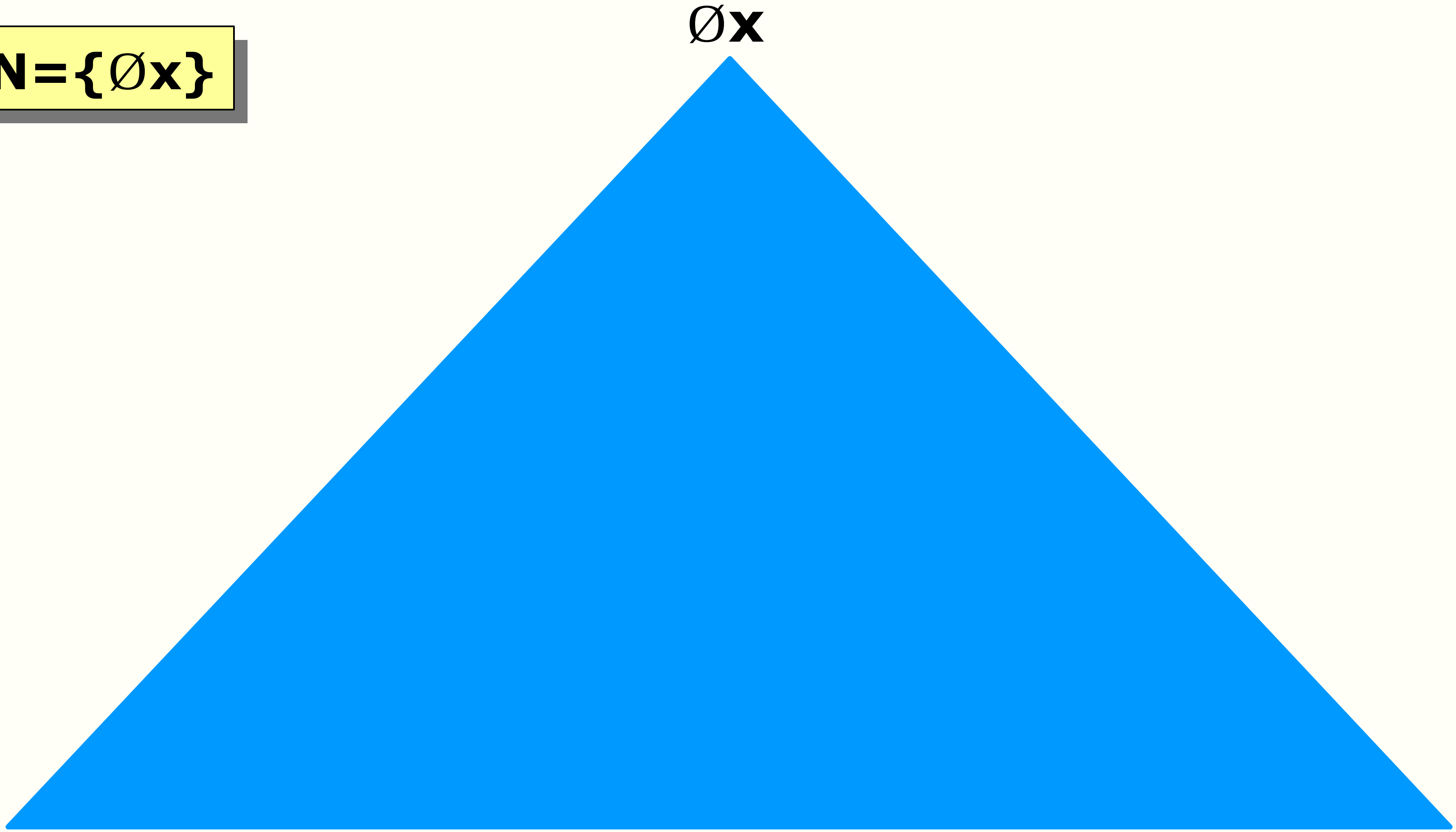
Extracting an Interpretation from a Branch



- ✗ A branch literal specifies the truth values for all its ground instances, unless there is a more specific literal specifying opposite truth values.

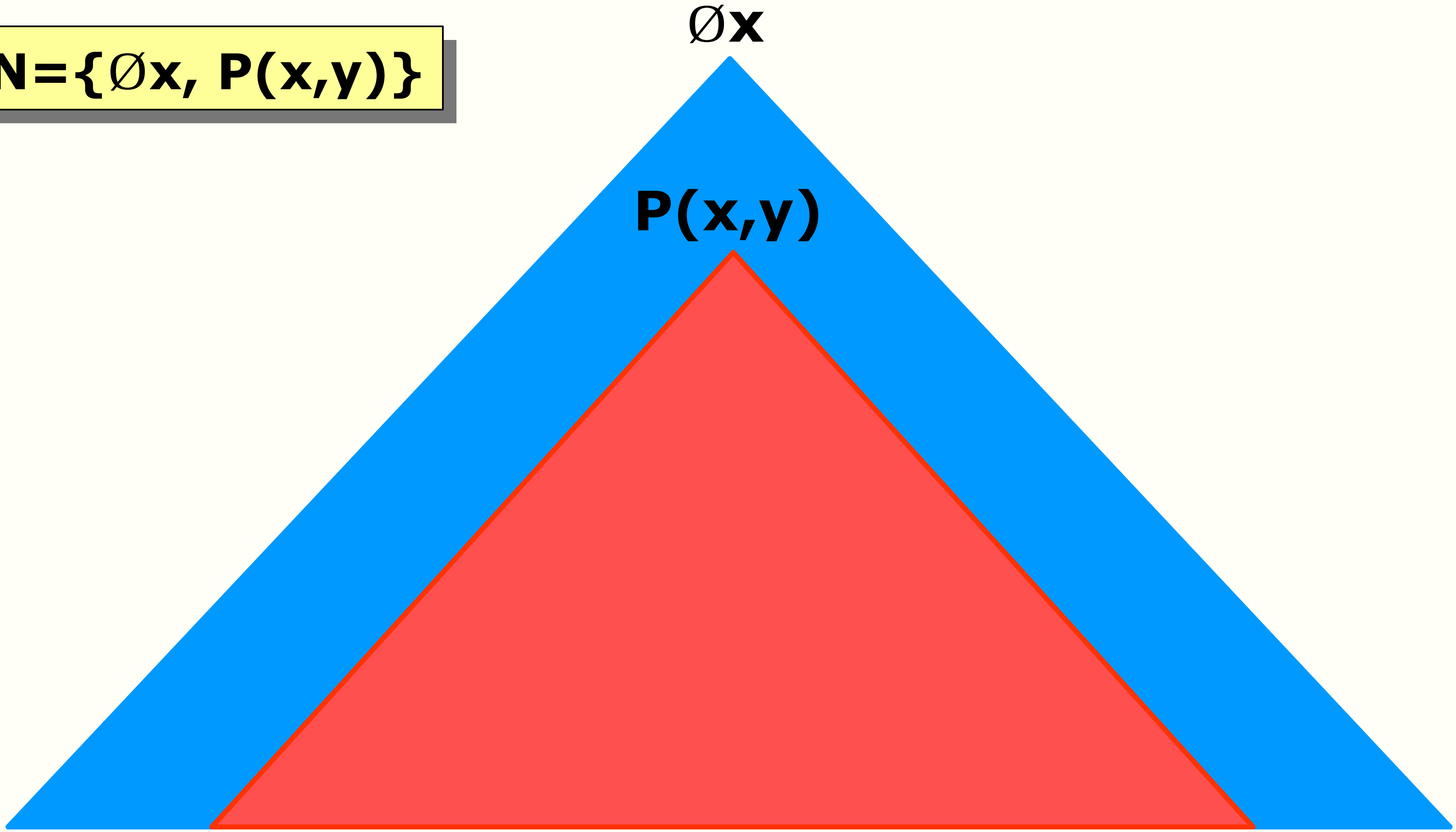
Branch (Literal Set) N

N={Øx}



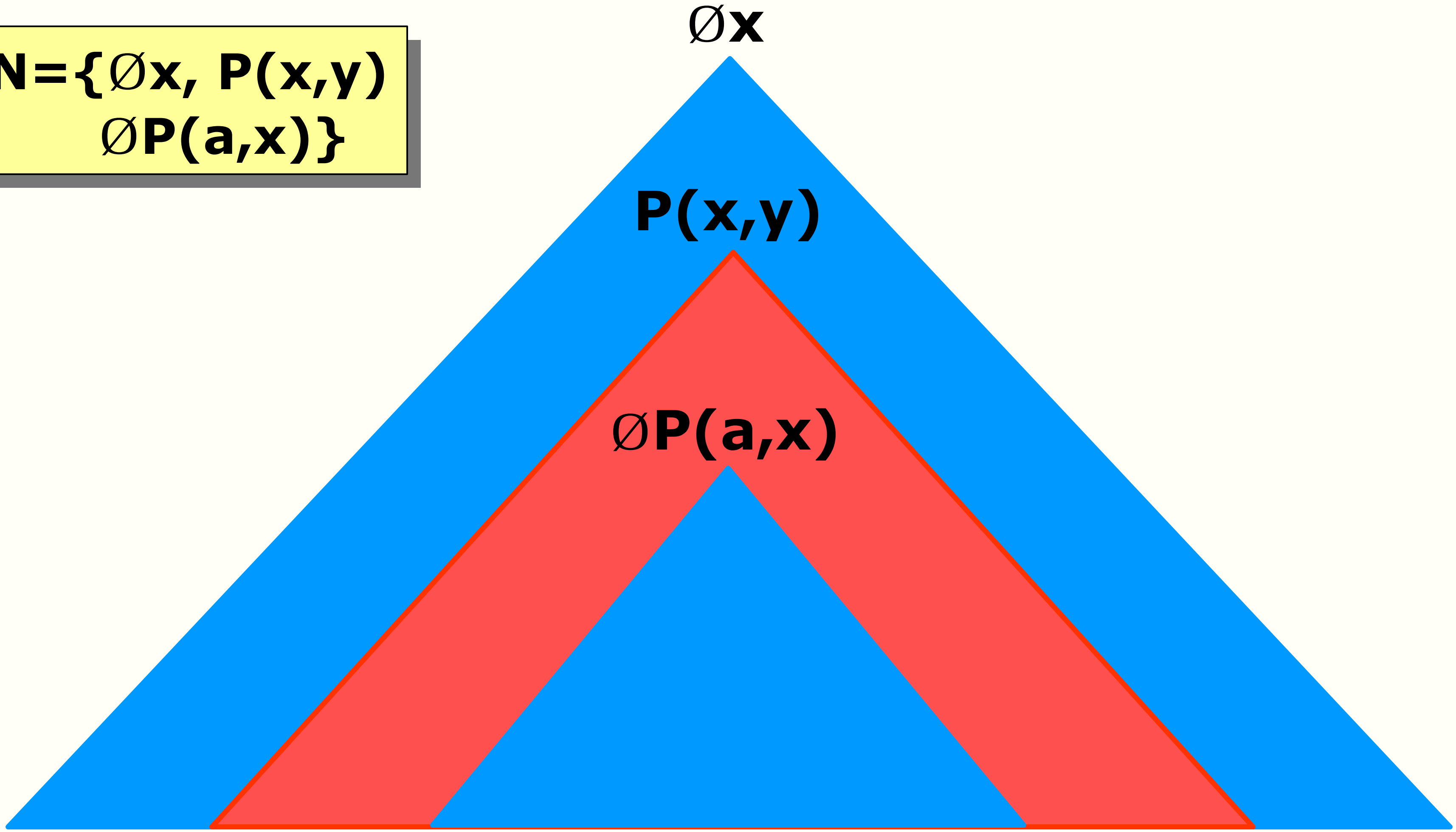
Branch (Literal Set) N

N={Øx, P(x,y)}



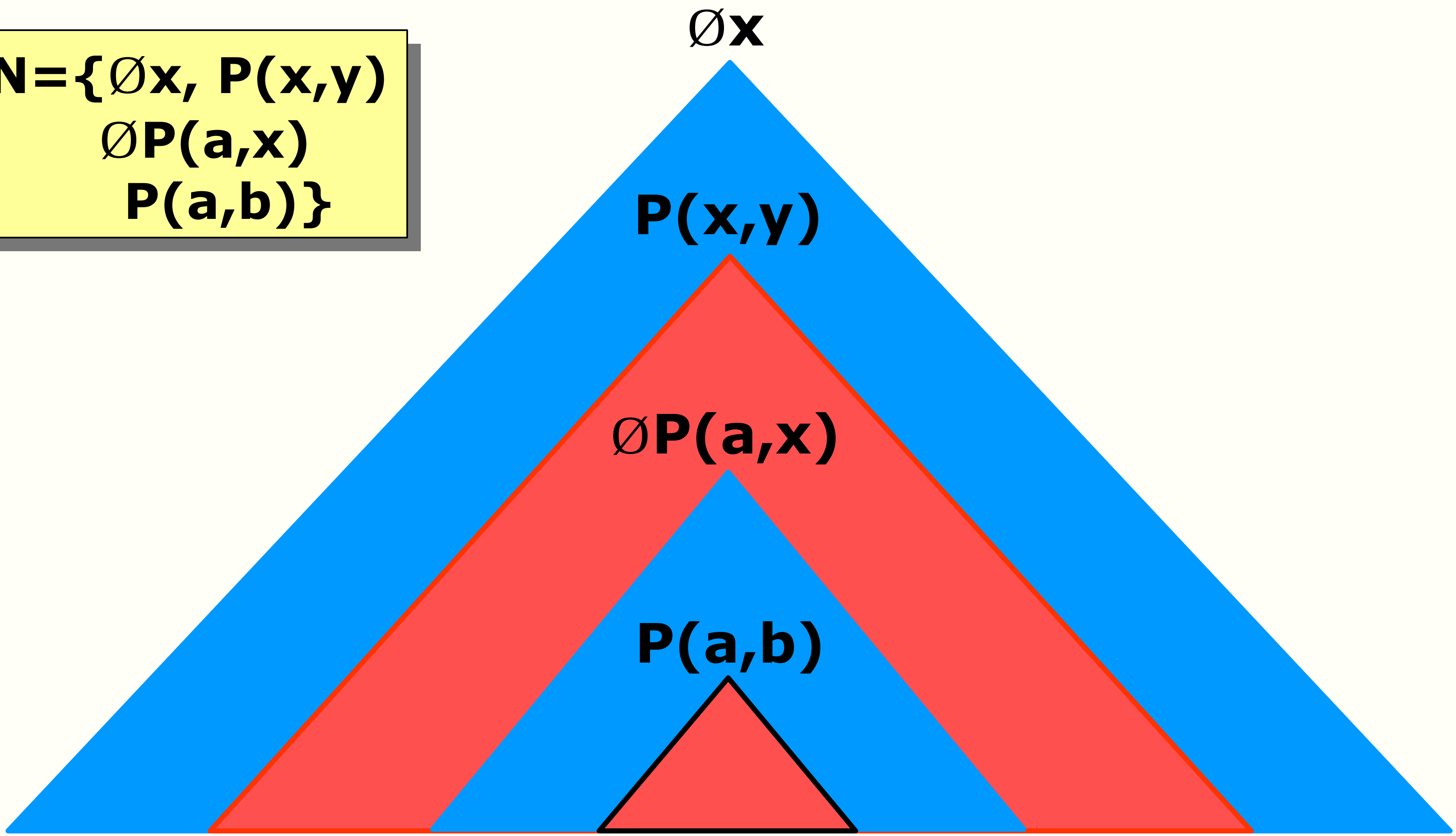
Branch (Literal Set) N

N={Øx, P(x,y), ØP(a,x)}



Branch (Literal Set) N

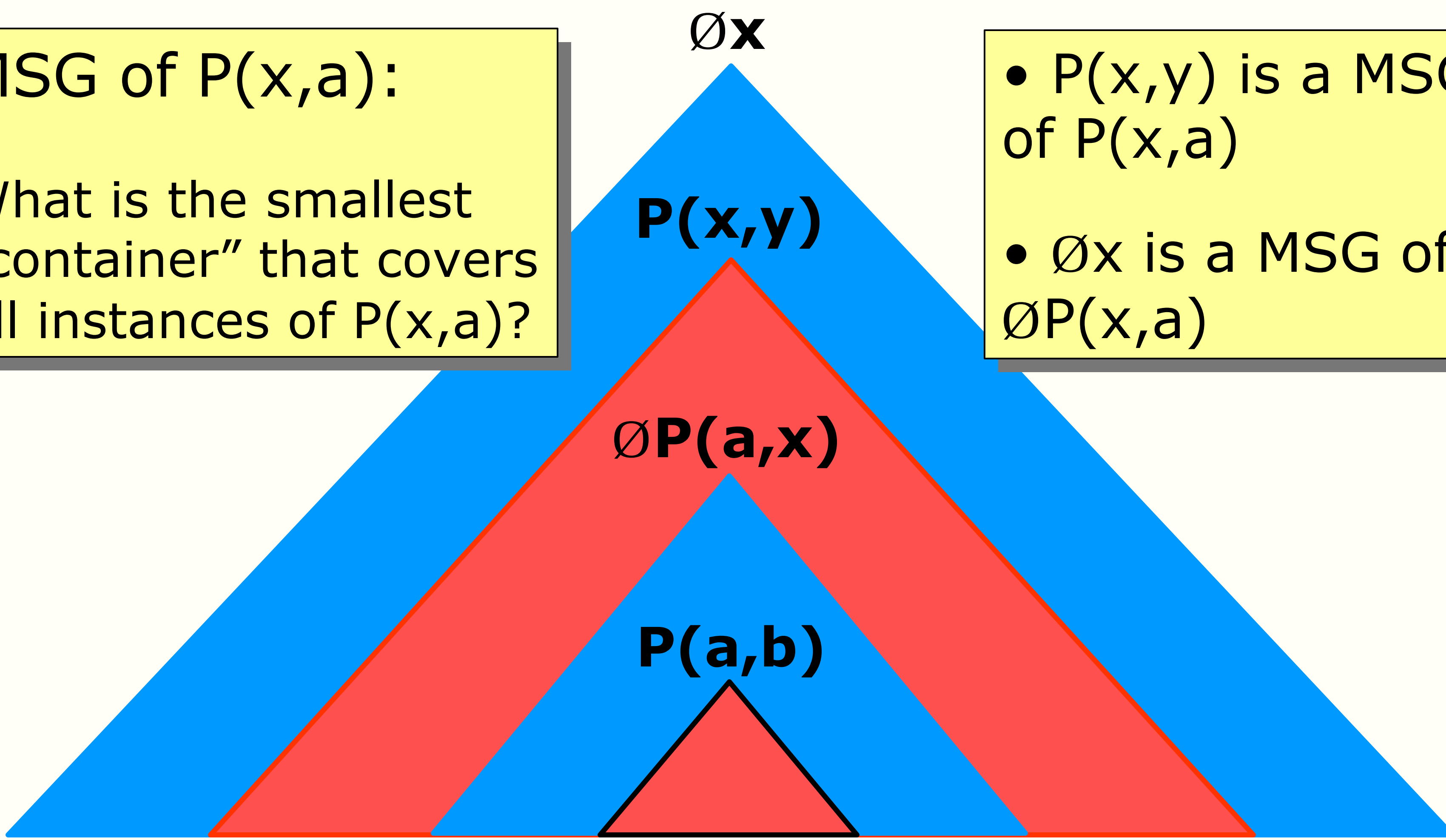
N={Øx, P(x,y)
ØP(a,x)
P(a,b)}



Most Specific Generalization of L in N

MSG of P(x,a):

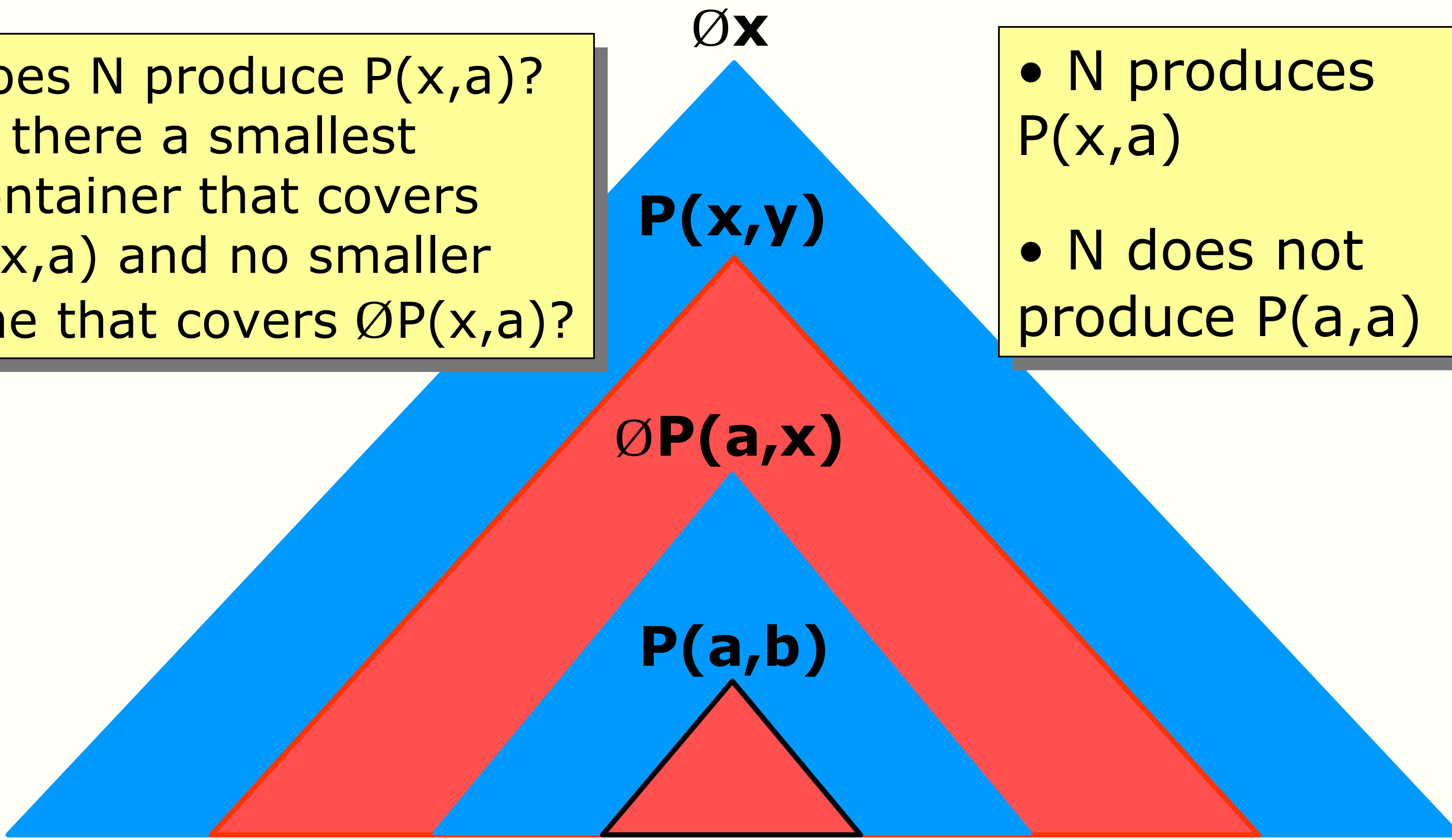
What is the smallest "container" that covers all instances of P(x,a)?



- P(x,y) is a MSG of P(x,a)
- Øx is a MSG of ØP(x,a)

Branch N Produces L

Does N produce P(x,a)? Is there a smallest container that covers P(x,a) and no smaller one that covers ØP(x,a)?



- N produces P(x,a)
- N does not produce P(a,a)

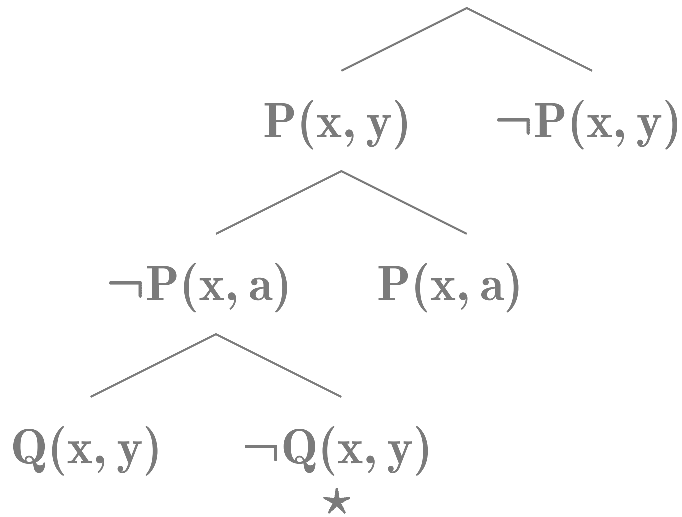
Interpretation

- Let N be a branch, [[N]] are all the ground literals that are produced by the branch
- If N includes $\neg x$ then [[N]] is complete (includes either L or $\neg L$ for every ground L)
- If [[N]] includes both L and $\neg L$ then N is inconsistent
- If N is complete and consistent it is an interpretation! ☺

Interpretation

- Let N be a branch, [[N]] are all the ground literals that are produced by the branch
- If N includes ¬x then [[N]] is complete (includes either L or ¬L for every ground L)
- If [[N]] includes both L and ¬L then N is inconsistent
- If N is complete and consistent it is an interpretation! ☺
- **"N produces L" does not coincide with "[[N]] models L" for non-ground literals!**

First-Order Semantic Trees



Issues:

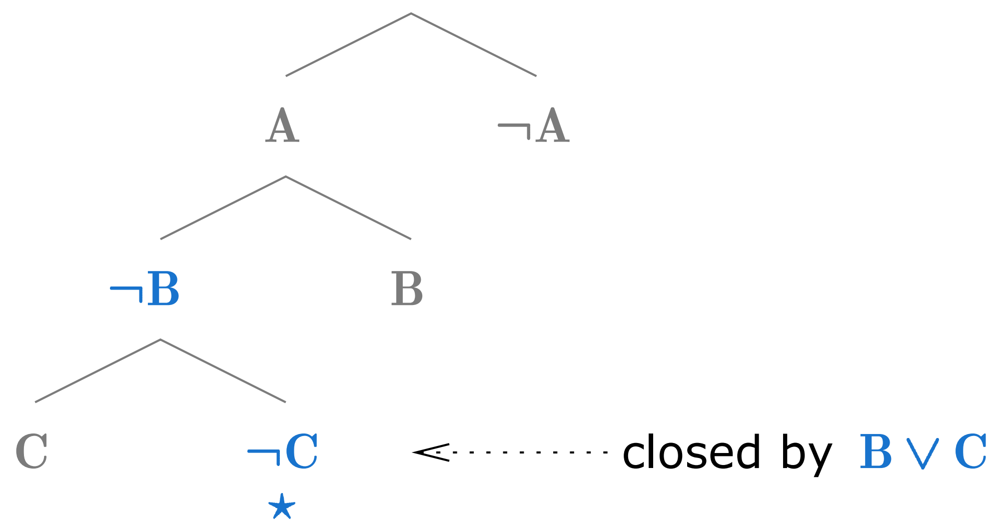
- ✗ How are variables treated?

  (a) Universal, as in Resolution?, (b) Rigid, as in Tableaux? (c) Schema!
- ✗ How to extract an interpretation from a branch? ✔
- ✗ **When is a branch closed?**
- ✗ How to construct such trees (calculus)?

Calculus: Branch Closure

Purpose: Determine if branch elementary contradicts an input clause.

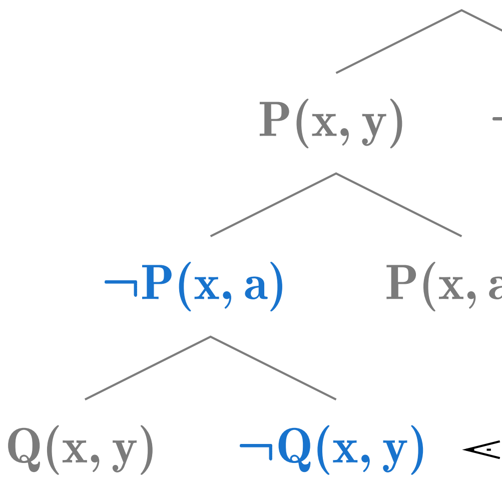
Propositional case:



Calculus: Branch Closure

Purpose: Determine if branch elementary contradicts an input clause.

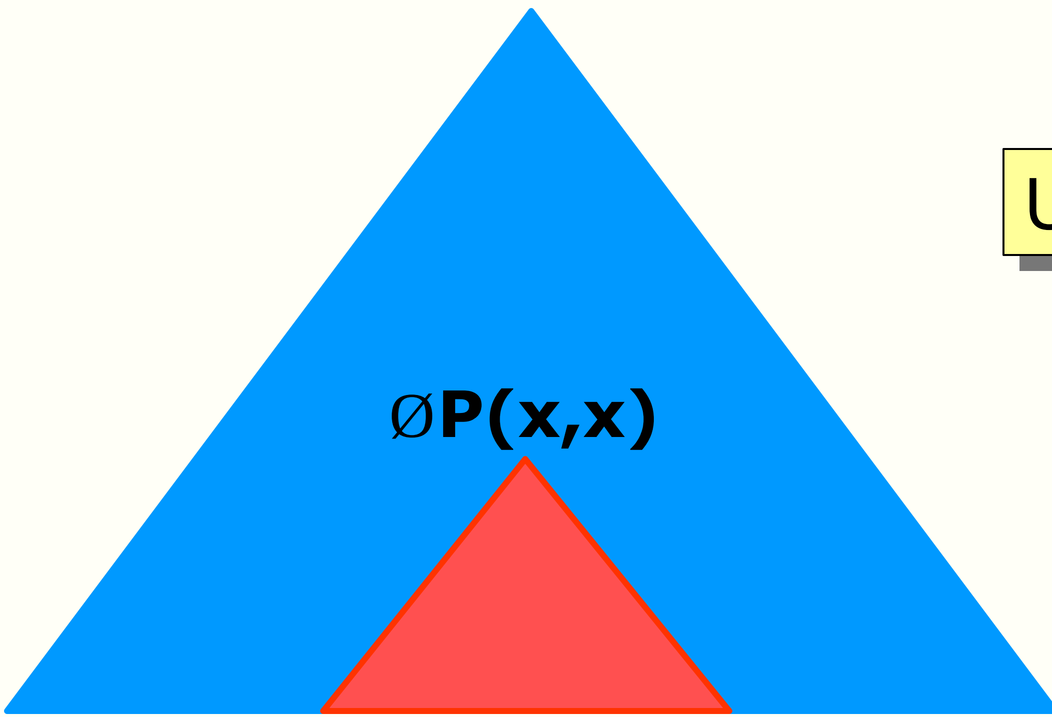
FDPLL case:



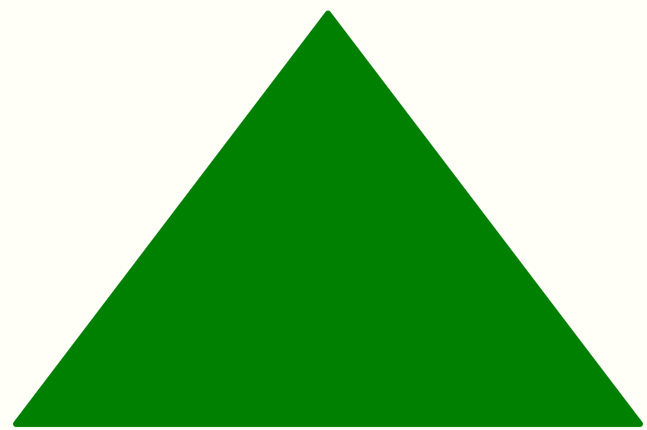
closed by $P(x,y) \lor Q(x,x)$?

Branch N Closed by C

Branch N: {Øx,…, ØP(x,x)}



Unit Clause C: P(x,y)

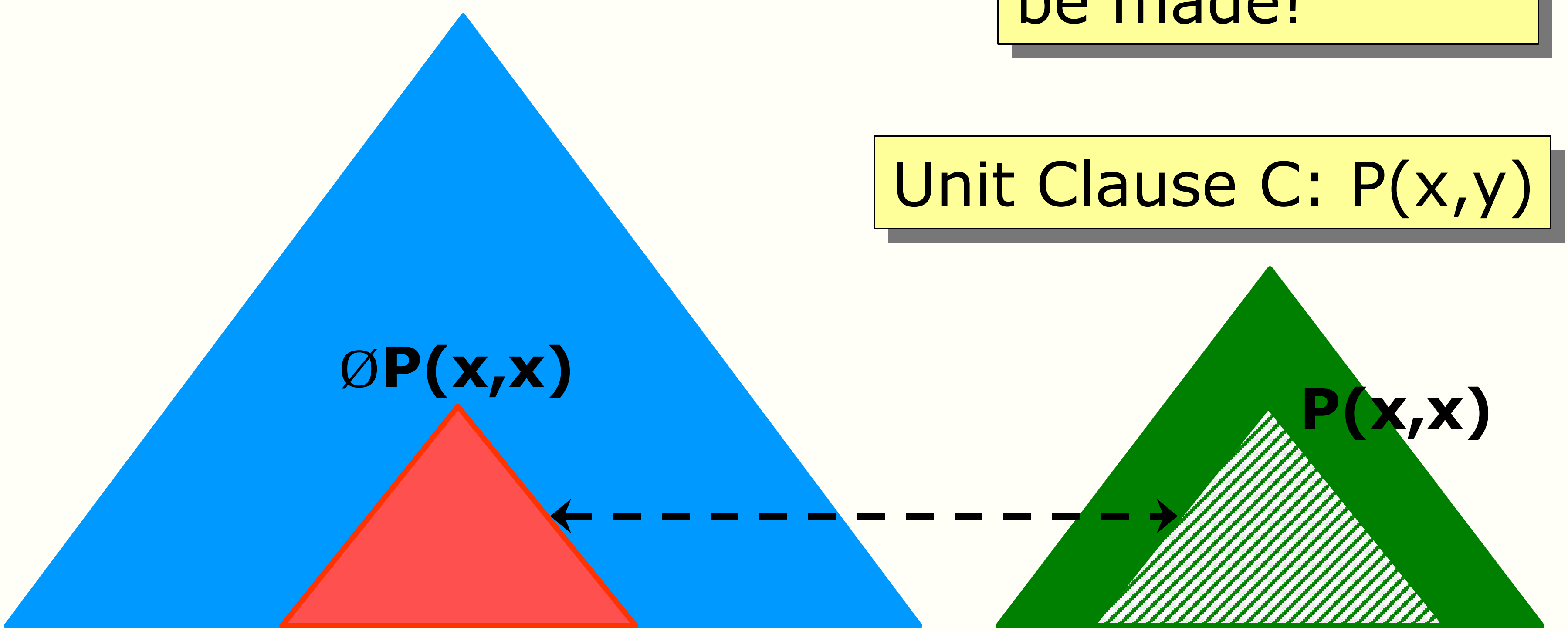


Branch N Closed by C

Branch N: {Øx,…, ØP(x,x)}

No progress can be made!

Unit Clause C: P(x,y)

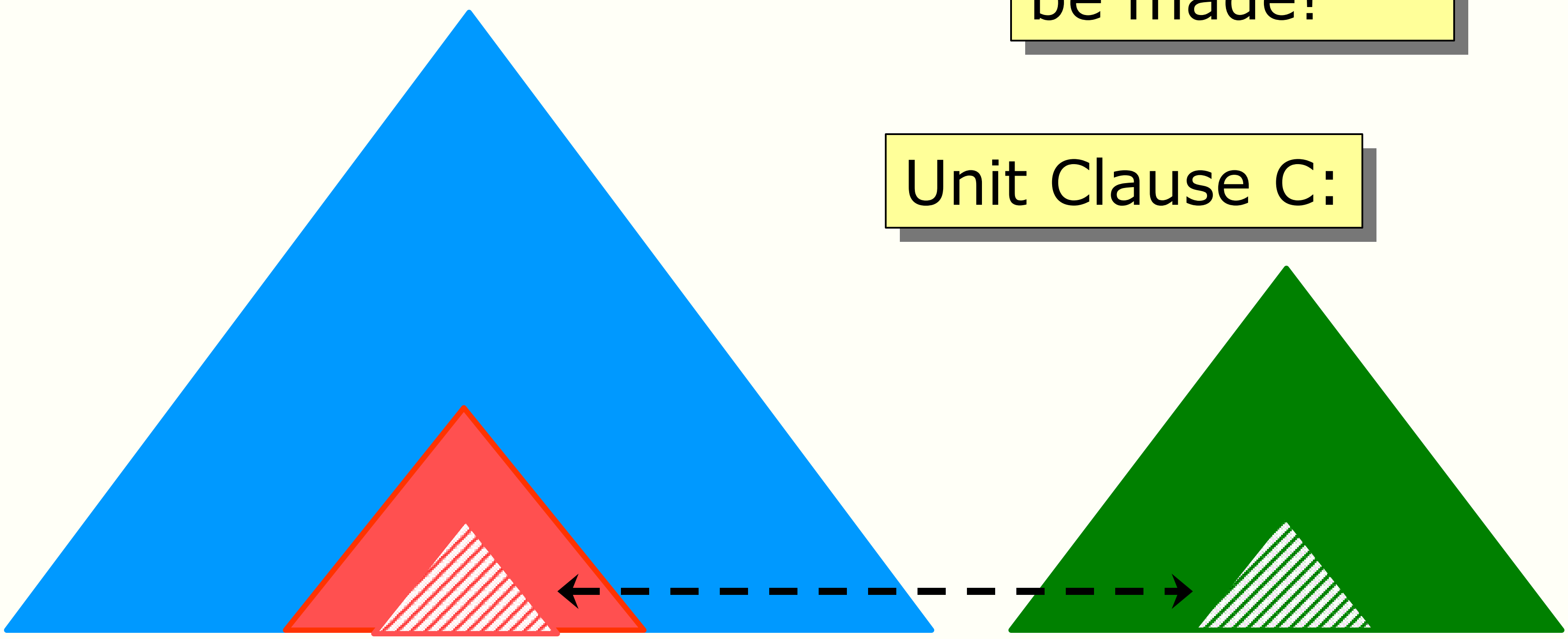


Repair Branch N (not Closed by C)

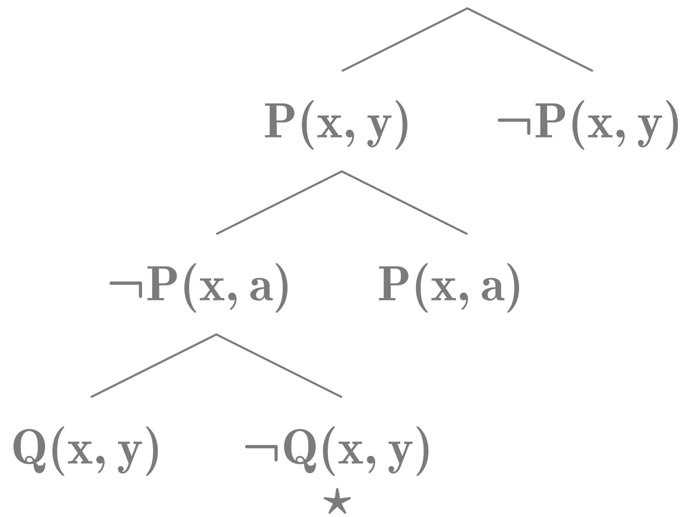
Branch N: {Øx,…}

Progress **can** be made!

Unit Clause C:



First-Order Semantic Trees



Issues:

- ✗ How are variables treated?

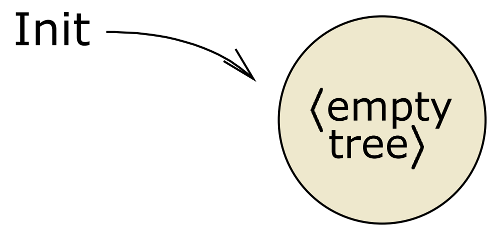
  (a) Universal, as in Resolution?, (b) Rigid, as in Tableaux? (c) Schema!
- ✗ How to extract an interpretation from a branch? ✔
- ✗ When is a branch closed? ✔
- ✗ **How to construct such trees (calculus)?**

FDPLL Calculus

Input: a clause set $\mathcal{S}$

Output: "unsatisfiable" or "satisfiable" (if terminates)

Note: Strategy much like in inner loop of propositional DPLL:

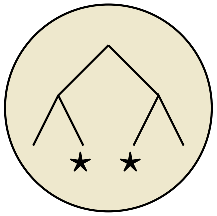


FDPLL Calculus

Input: a clause set $\mathcal{S}$

Output: “unsatisfiable” or “satisfiable” (if terminates)

Note: Strategy much like in inner loop of propositional DPLL:

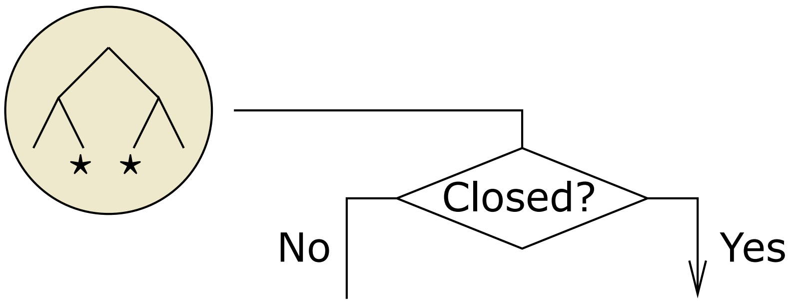


FDPLL Calculus

Input: a clause set $\mathcal{S}$

Output: "unsatisfiable" or "satisfiable" (if terminates)

Note: Strategy much like in inner loop of propositional DPLL:

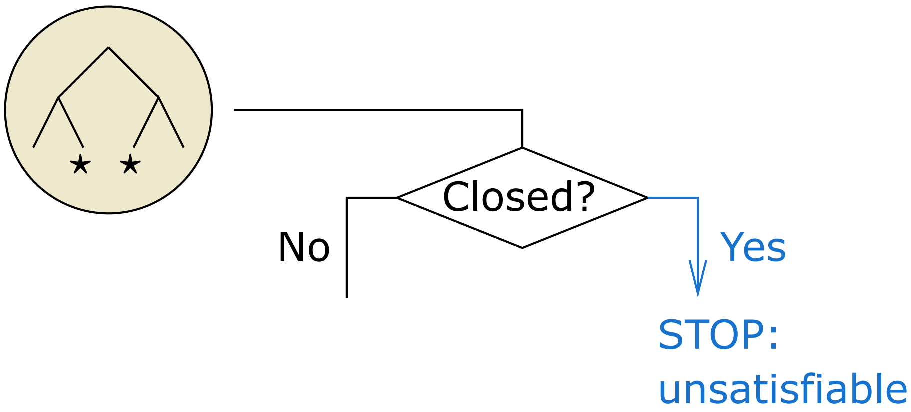


FDPLL Calculus

Input: a clause set $\mathcal{S}$

Output: "unsatisfiable" or "satisfiable" (if terminates)

Note: Strategy much like in inner loop of propositional DPLL:

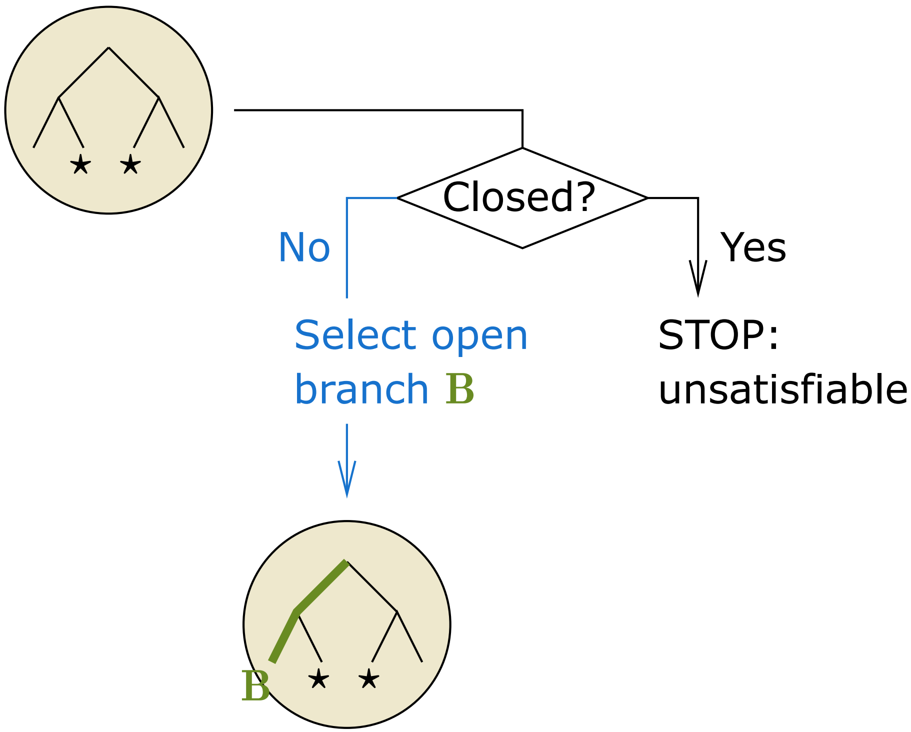


FDPLL Calculus

Input: a clause set $\mathcal{S}$

Output: “unsatisfiable” or “satisfiable” (if terminates)

Note: Strategy much like in inner loop of propositional DPLL:

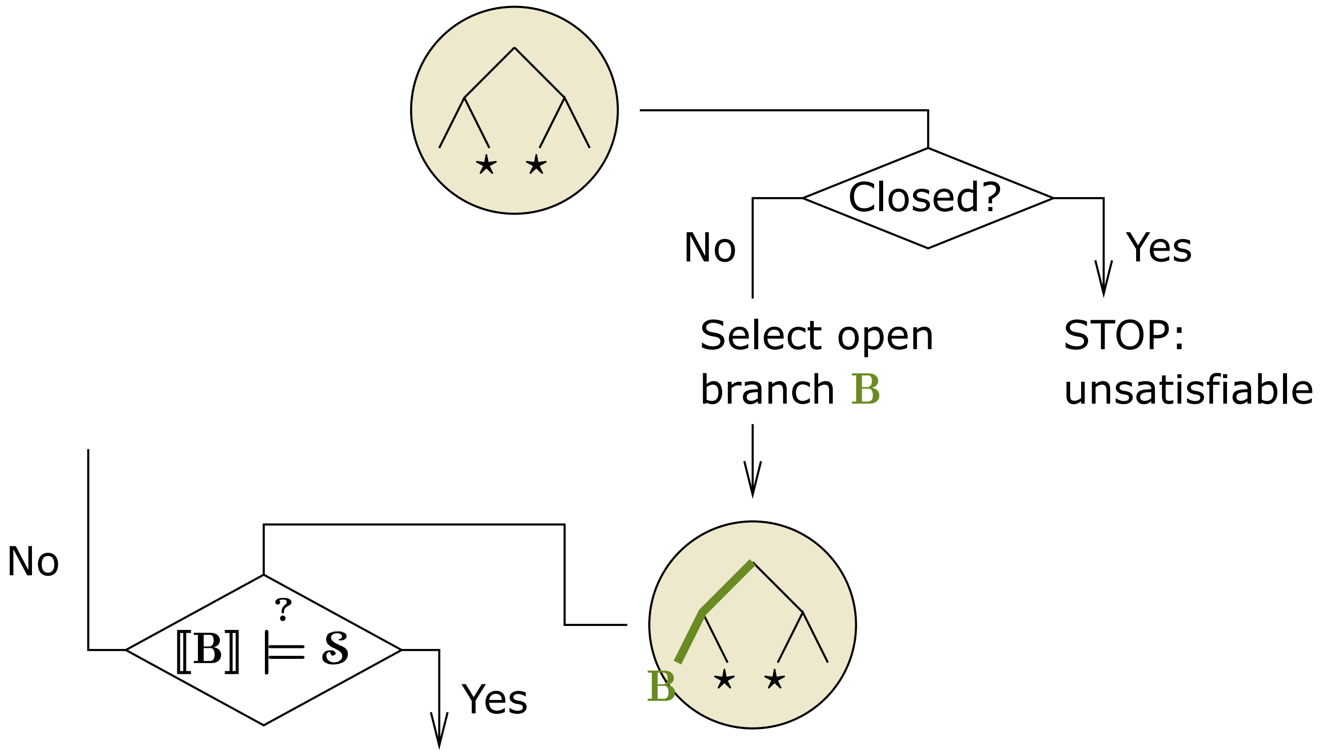


FDPLL Calculus

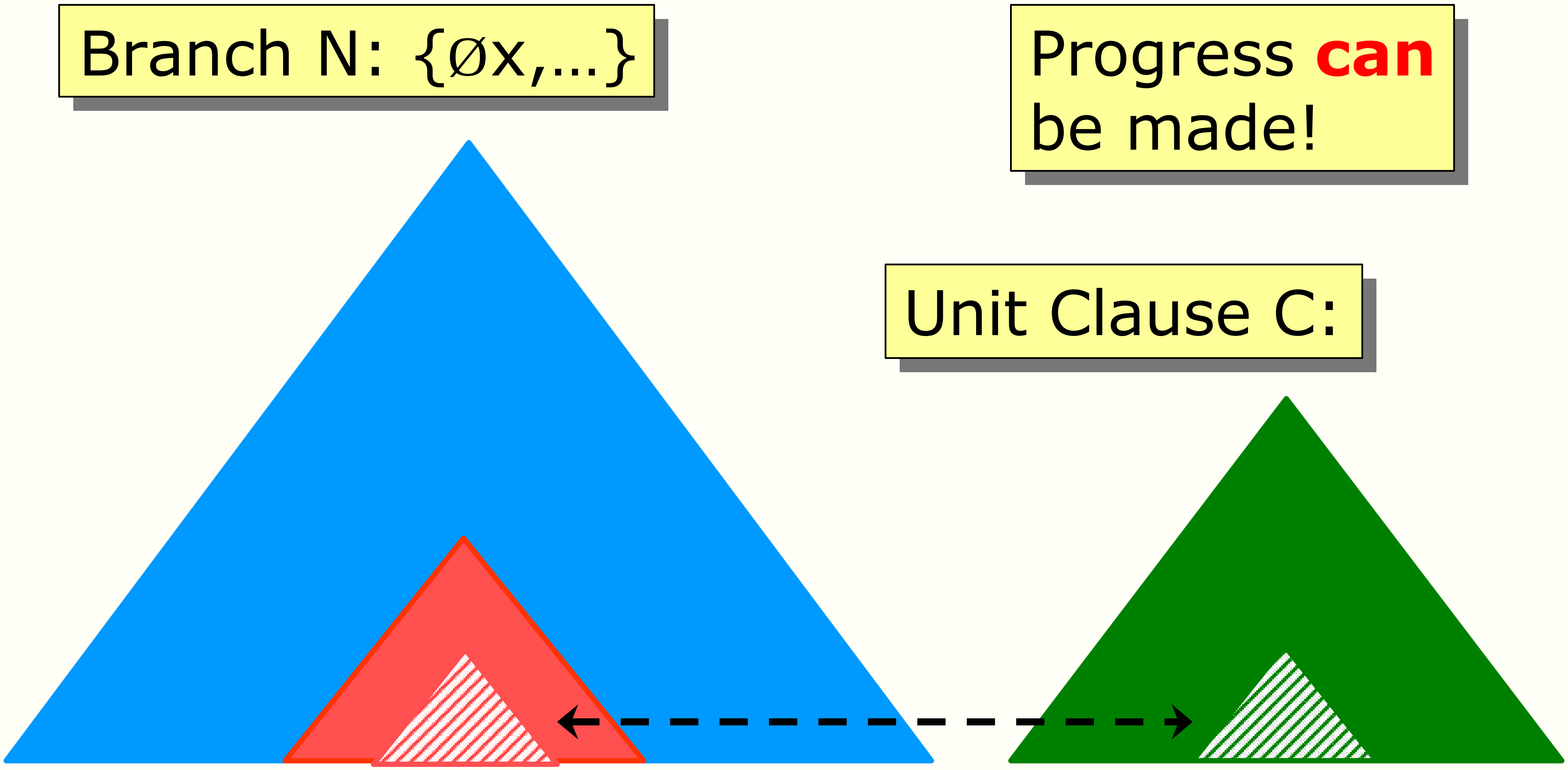
Input: a clause set $\mathcal{S}$

Output: "unsatisfiable" or "satisfiable" (if terminates)

Note: Strategy much like in inner loop of propositional DPLL:



Repair Branch N (not Closed by C)

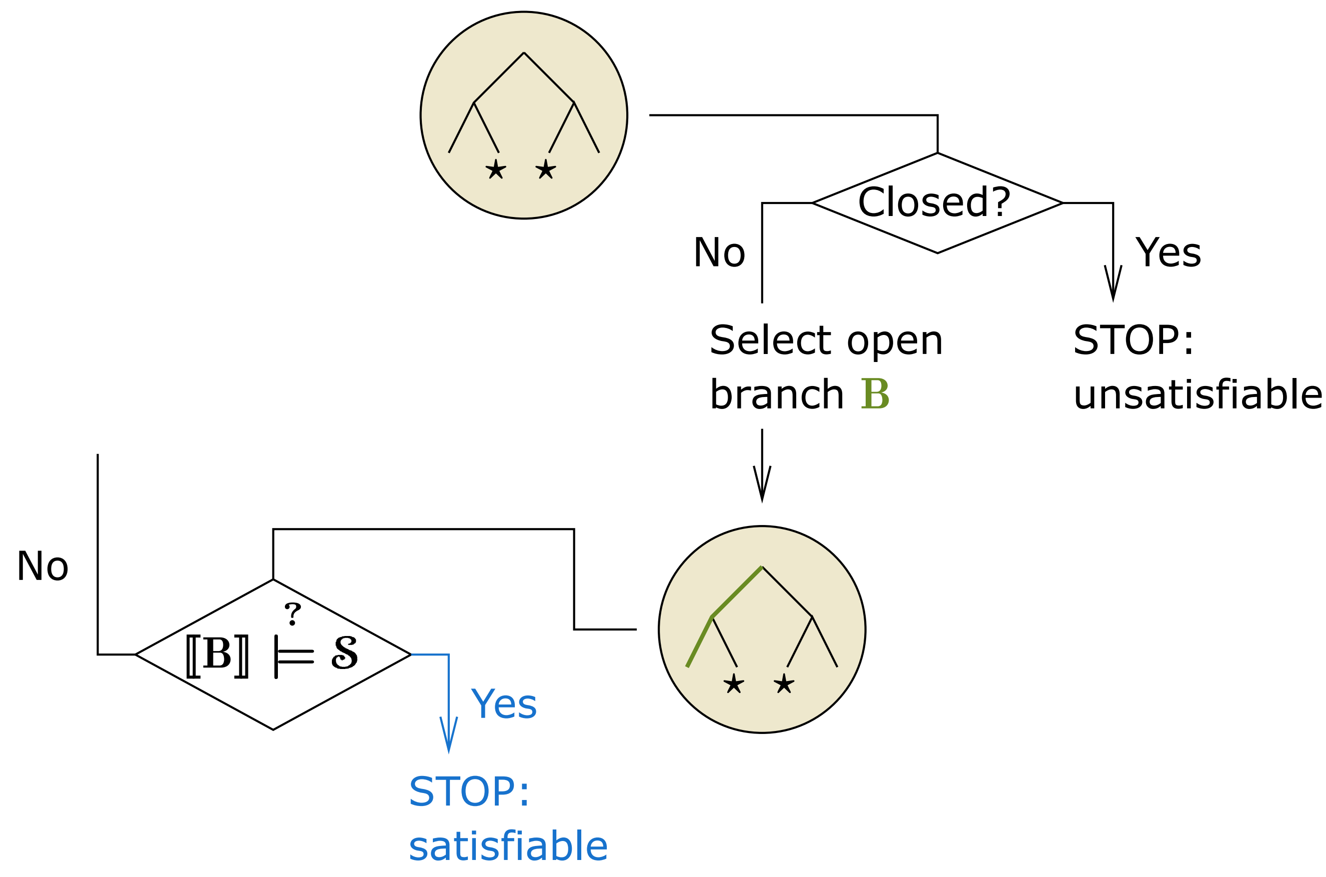


FDPLL Calculus

Input: a clause set $\mathcal{S}$

Output: "unsatisfiable" or "satisfiable" (if terminates)

Note: Strategy much like in inner loop of propositional DPLL:

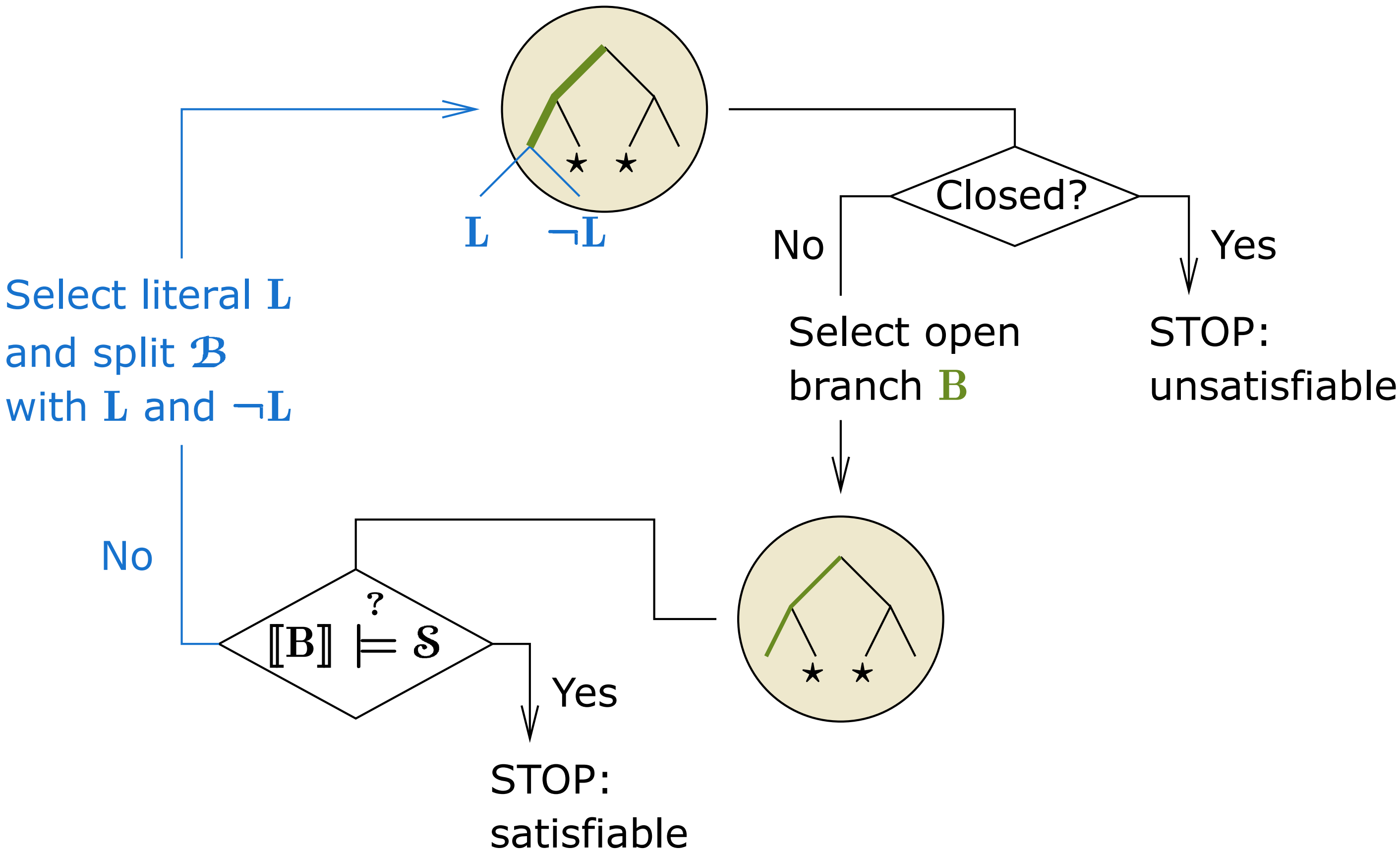


FDPLL Calculus

Input: a clause set $\mathcal{S}$

Output: “unsatisfiable” or “satisfiable” (if terminates)

Note: Strategy much like in inner loop of propositional DPLL:

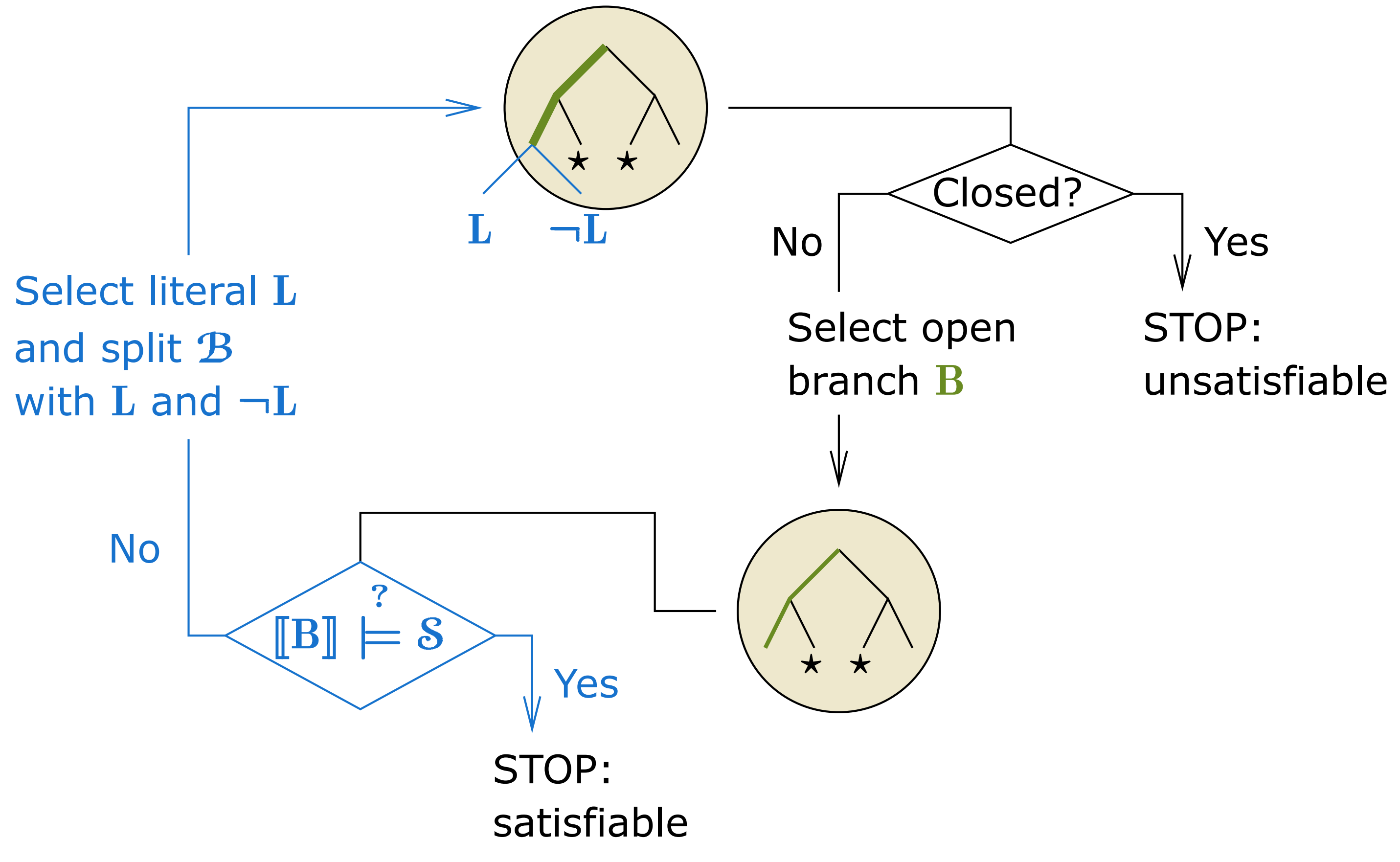


FDPLL Calculus

Input: a clause set $\mathcal{S}$

Output: "unsatisfiable" or "satisfiable" (if terminates)

Note: Strategy much like in inner loop of propositional DPLL:



Next: Testing $[\![\mathbf{B}]\!] \models \mathcal{S}$ and splitting

Calculus: The Splitting Rule

Purpose: Satisfy a clause that is currently "false"

$$\mathbf{P(y'', x'')}$$

$$\mathbf{\neg P(a, y')}$$

$$\mathbf{\neg P(a, b)}$$

Some clause from $\mathcal{S}$

$$\mathbf{P(x, y) \lor \neg P(y, x)}$$

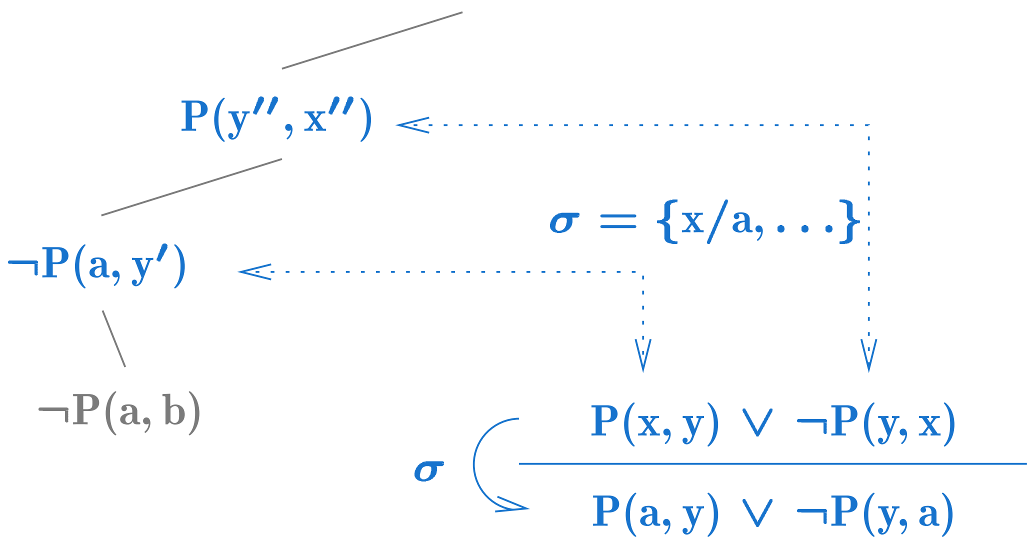
1.

2.

3.

Calculus: The Splitting Rule

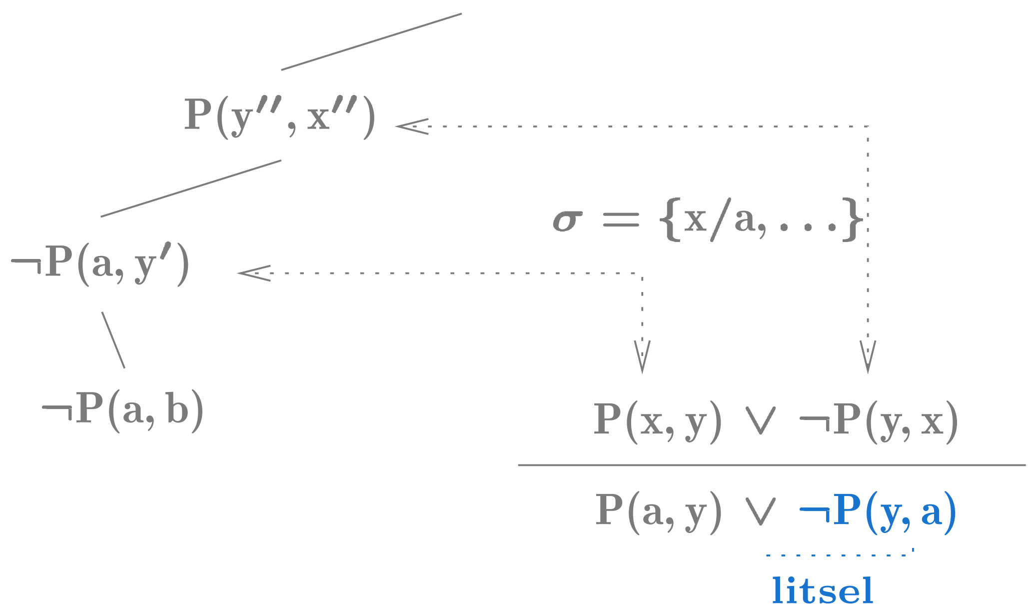
Purpose: Satisfy a clause that is currently “false”



1. Compute simultaneous most general unifier σ
2.
3.

Calculus: The Splitting Rule

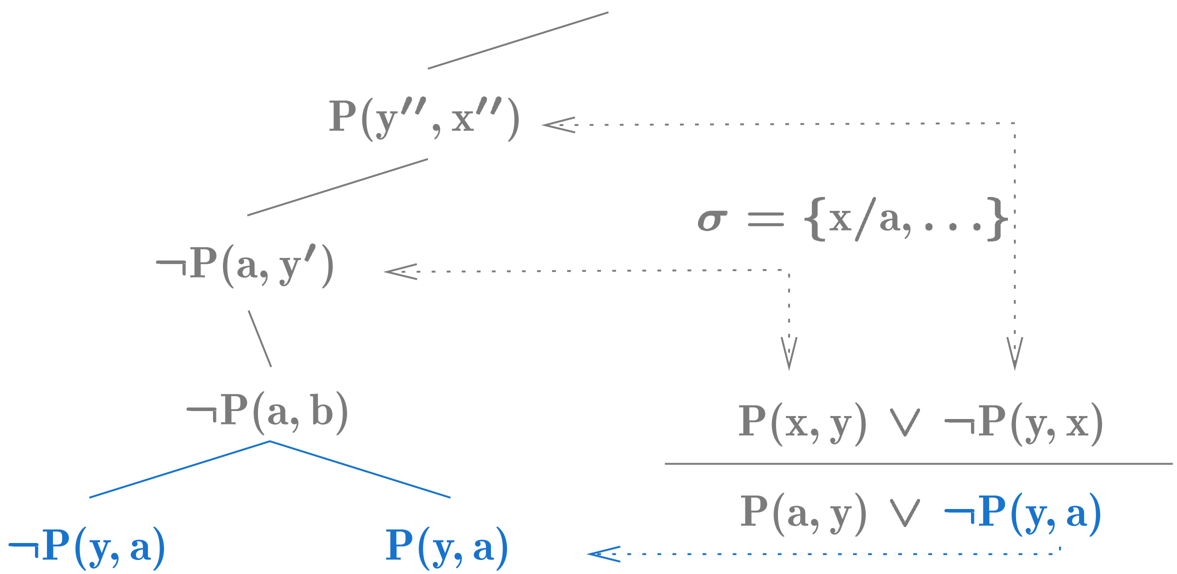
Purpose: Satisfy a clause that is currently "false"



1. Compute simultaneous most general unifier σ
2. Select from clause instance a literal not on branch
3.

Calculus: The Splitting Rule

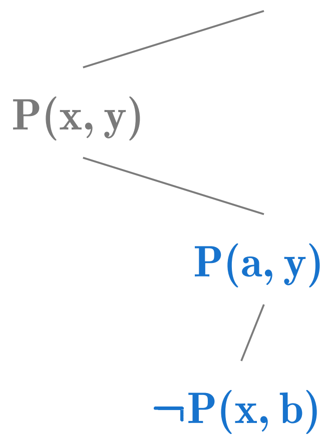
Purpose: Satisfy a clause that is currently "false"



1. Compute simultaneous most general unifier σ
2. Select from clause instance a literal not on branch
3. Split with this literal

Calculus: The Commit Rule

Purpose: Achieve consistency of interpretation associated to branch



Calculus: The Commit Rule

Purpose: Achieve consistency of interpretation associated to branch

$\mathbf{P(x, y)}$

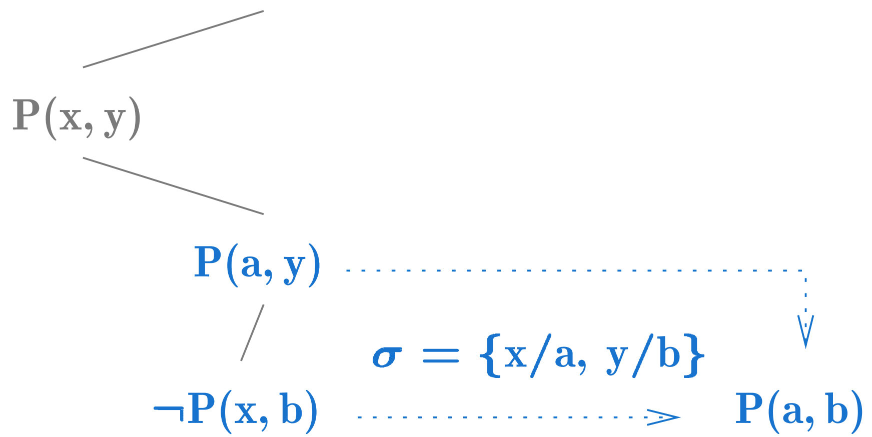
$\mathbf{P(a, y)}$

$\mathbf{\neg P(x, b)}$

Problem: $\mathbf{\{P(a, b), \neg P(a, b)\}}$ is inconsistent!

Calculus: The Commit Rule

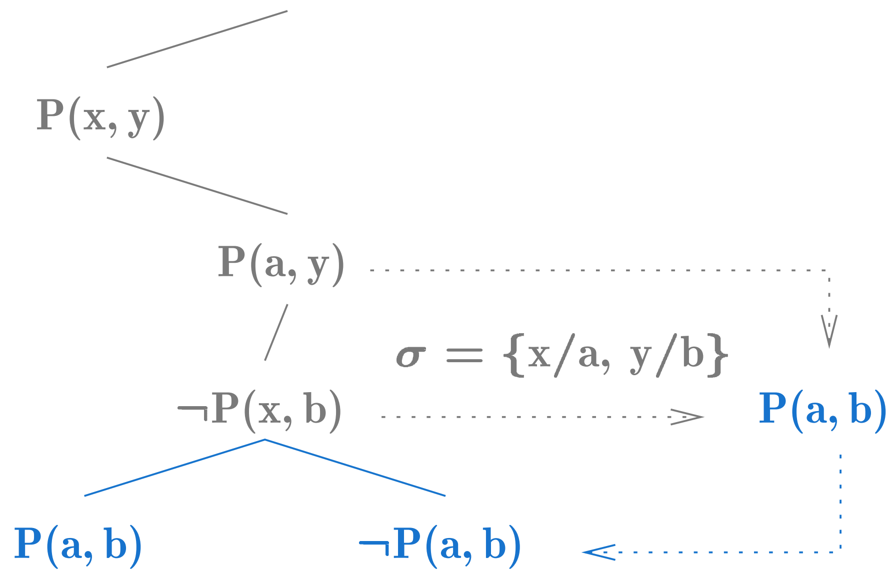
Purpose: Achieve consistency of interpretation associated to branch



1. Compute a MGU σ of branch literals with opposite sign
2.

Calculus: The Commit Rule

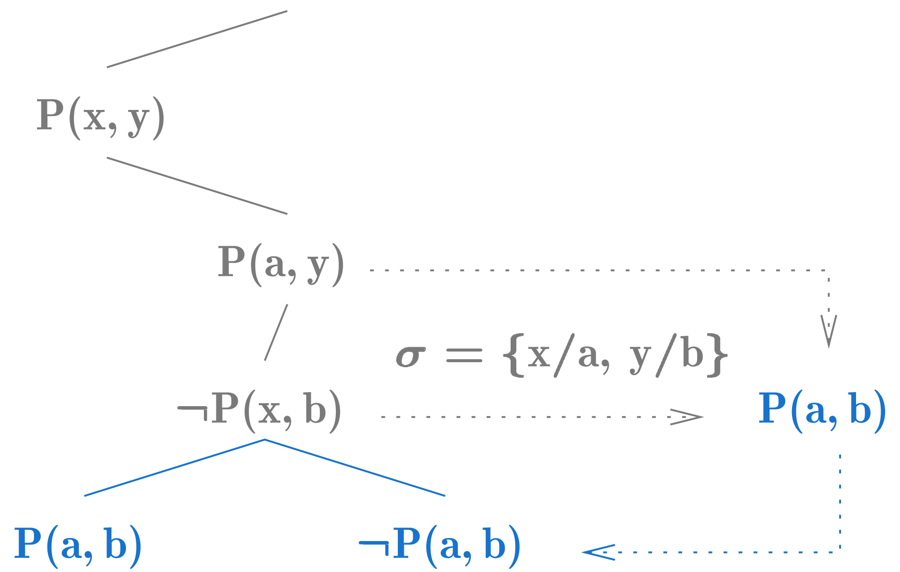
Purpose: Achieve consistency of interpretation associated to branch



1. Compute a MGU σ of branch literals with opposite sign
2. Split with instance, if not on branch

Calculus: The Commit Rule

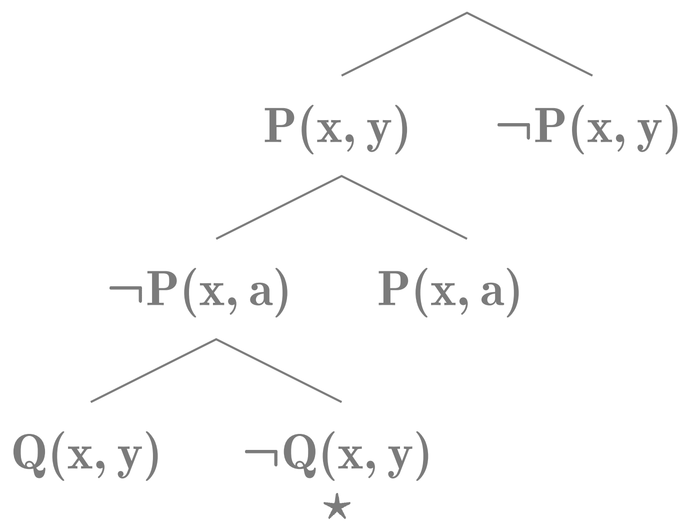
Purpose: Achieve consistency of interpretation associated to branch



1. Compute a MGU σ of branch literals with opposite sign
2. Split with instance, if not on branch

Now have removed the inconsistency

First-Order Semantic Trees



Issues:

- ✗ How are variables treated?
  (a) Universal, as in Resolution?, (b) Rigid, as in Tableaux? (c) Schema!
- ✗ How to extract an interpretation from a branch? ✔
- ✗ When is a branch closed? ✔
- ✗ How to construct such trees (calculus)? ✔

Soundness and Completeness

- Soundness:

  If FDPLL derives a refutation for C
  then C is unsatisfiable.

- Completeness ***** :

  If C is unsatisfiable
  then FDPLL derives a refutation.

- FOL is semi-decidable:
  - FDPLL is not guaranteed to terminate with a model of C when there is one.

Discussion

- Why is this lifting from propositional to FOL important?
  - Propositional logic is limited comparing to FOL: FOL allows infinite domains, unnamed constants,…
  - FOL is much more concise: propositional problems might even be solved more efficient when represented in FOL.
- FOL is semi-decidable so why do we care anyway?
  - There are decidable fragments such as the B-S class, Description Logics, …
- Relation to other proof calculi: Resolution, Semantic Tableaux,…
  - Sound and Complete, but how about proof length?
- So, is this efficient after all? How does it compare to other FOL theorem provers?