# A Tree-based Modular SMT Solver

**Georg Schadler**
**Georg Hofferek**

# Outline

- Motivation
- Proof structure, requirements & properties
- Implementation & example
- Theory checks & interpolation
- Outlook & conclusion

# Motivation

# Motivation

- Synthesize **multiple Boolean control signals** e.g. for a pipelined processor.
- Specification given as a **quantified first-order formula**.
- **Uninterpreted functions** to abstract specification

# Quantified Formula

$$\Psi = \forall\, inputs, states\,.\ \exists\, controlsignals\,.\ \forall\, auxvars\,.\ \Phi$$

- $inputs, states$: Stuff that choice of control signals depends on
- $controlsignals$: Unknown → To be synthesized
- $auxvars$: Stuff that control signals *do not depend on*
- $\Phi$: Correctness axiom

# Formula Expansion

$$\forall \vec{a}\ \exists c_0, c_1\ \forall \vec{b}\ .\ \Phi\left(\vec{a}, \vec{b}, c_0, c_1\right) = \top$$

- Expansion of $\exists$
- Renaming of $\vec{b}$
- Negation

$$\neg\Phi\left(\vec{a}, \vec{b}_{00}, 0,0\right) \wedge \Phi\left(\vec{a}, \vec{b}_{10}, 1,0\right) \wedge \Phi\left(\vec{a}, \vec{b}_{01}, 0,1\right) \wedge \Phi\left(\vec{a}, \vec{b}_{11}, 1,1\right) = \bot$$

**“Partitions”**: $\phi_{00}, \phi_{01}, \phi_{10}, \phi_{11}$

# Motivation Recap

1. Construct unsatisfiable SMT formula from specification and compute proof
2. Craig interpolation to compute multiple coordinated interpolants.
3. Interpolants implement the Boolean control signals.

More Information:
Hofferek et al., FMCAD 2013
Hofferek & Bloem, MEMOCODE 2011

# Refutation Proof

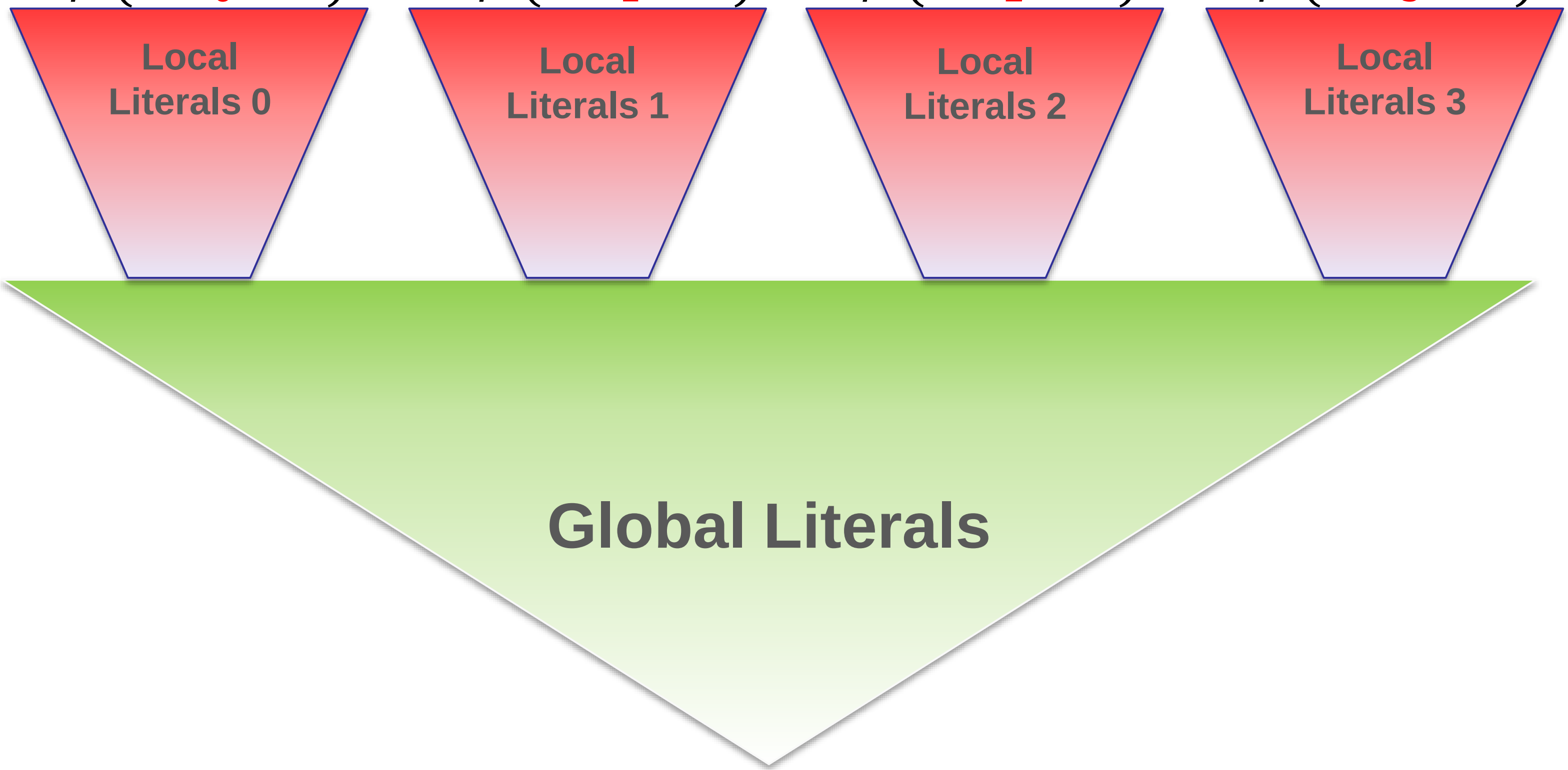
- Proof requires two properties:
  - **Local-first**

    Local literals are resolved before global literals
  - **Colorable**

    No literals or leaves with symbols from two partitions

# Proof Requirements

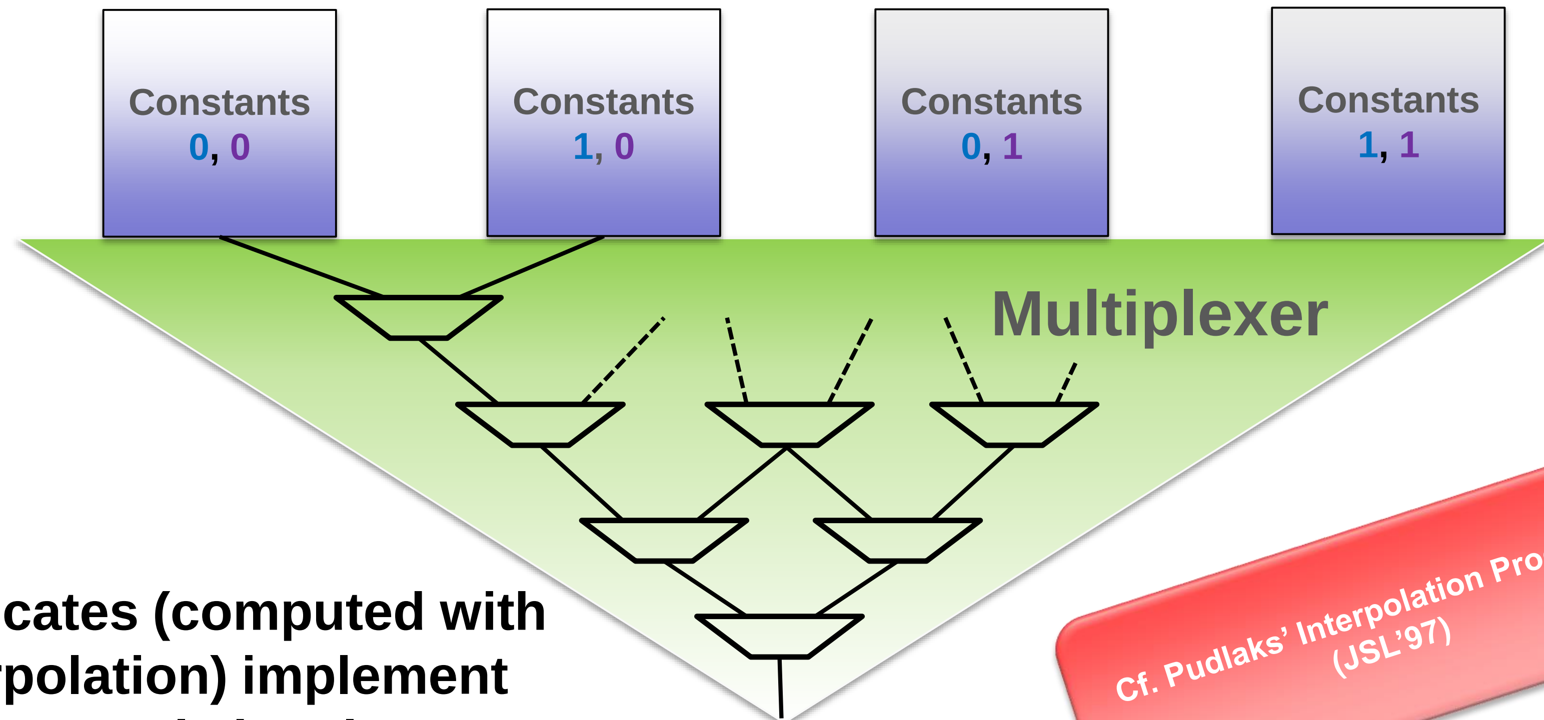
$$\neg\varphi\left(\vec{a},\vec{b}_0,0,0\right) \wedge \neg\varphi\left(\vec{a},\vec{b}_1,1,0\right) \wedge \neg\varphi\left(\vec{a},\vec{b}_2,0,1\right) \wedge \neg\varphi\left(\vec{a},\vec{b}_3,1,1\right)$$




$$I_0 \quad \perp \quad I_1$$

# Proof Requirements

$$\neg\varphi\left(\vec{a},\vec{b}_0,0,0\right) \wedge \neg\varphi\left(\vec{a},\vec{b}_1,1,0\right) \wedge \neg\varphi\left(\vec{a},\vec{b}_2,0,1\right) \wedge \neg\varphi\left(\vec{a},\vec{b}_3,1,1\right)$$




**Certificates (computed with interpolation) implement control signals**

$I_0$ $\bot$ $I_1$

Cf. Pudlaks' Interpolation Procedure (JSL'97)

# Colorability

Partitions ≈ Colors:

$$\neg\Phi_{00}(\vec{a},\vec{b}_{00}) \wedge \neg\Phi_{10}(\vec{a},\vec{b}_{10}) \wedge \neg\Phi_{01}(\vec{a},\vec{b}_{01}) \wedge \neg\Phi_{11}(\vec{a},\vec{b}_{11})$$

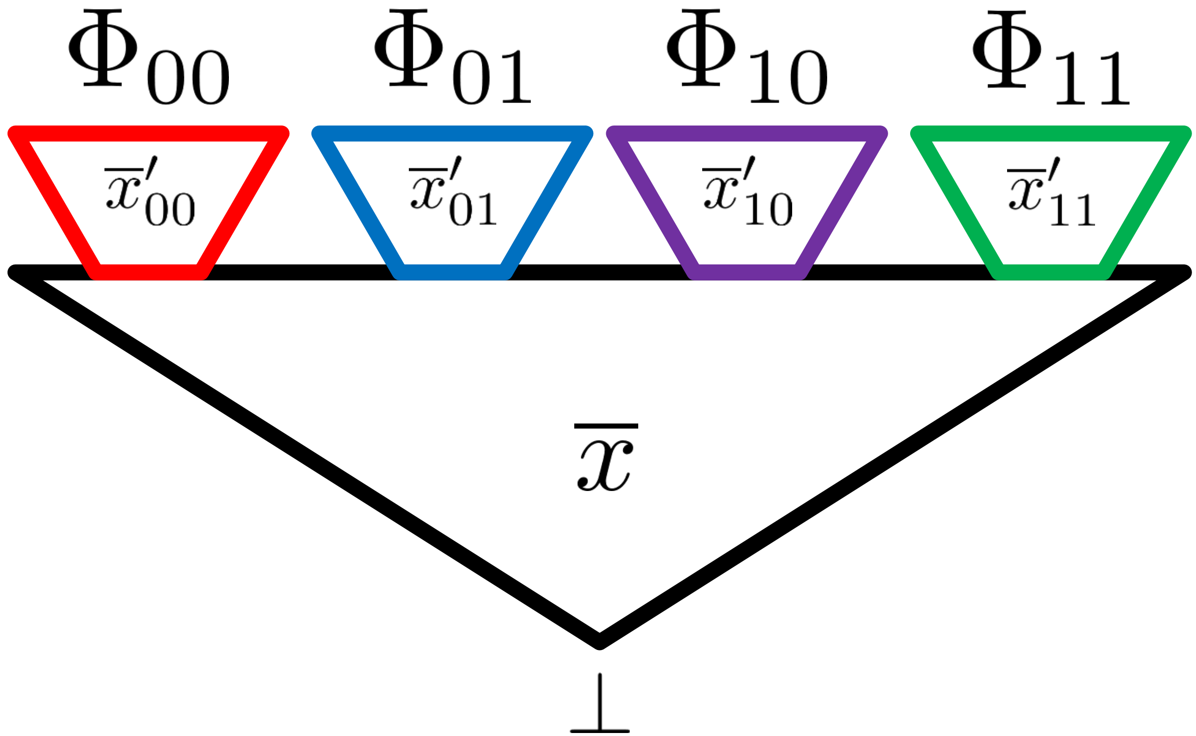
Local Symbols: $\vec{b}_{00}$, $\vec{b}_{10}$, $\vec{b}_{01}$, $\vec{b}_{11}$ (colored)

Global Symbols: $\vec{a}$ (colorless)

Colorable: $(\boldsymbol{x} = \boldsymbol{y})$, $(\boldsymbol{u} = \boldsymbol{v})$, $(\boldsymbol{w} = z)$

Non-colorable: $(\boldsymbol{x} = \boldsymbol{u})$

# Colorability




**No literals or leaves with symbols from two partitions**

# Implementation

# Implementation

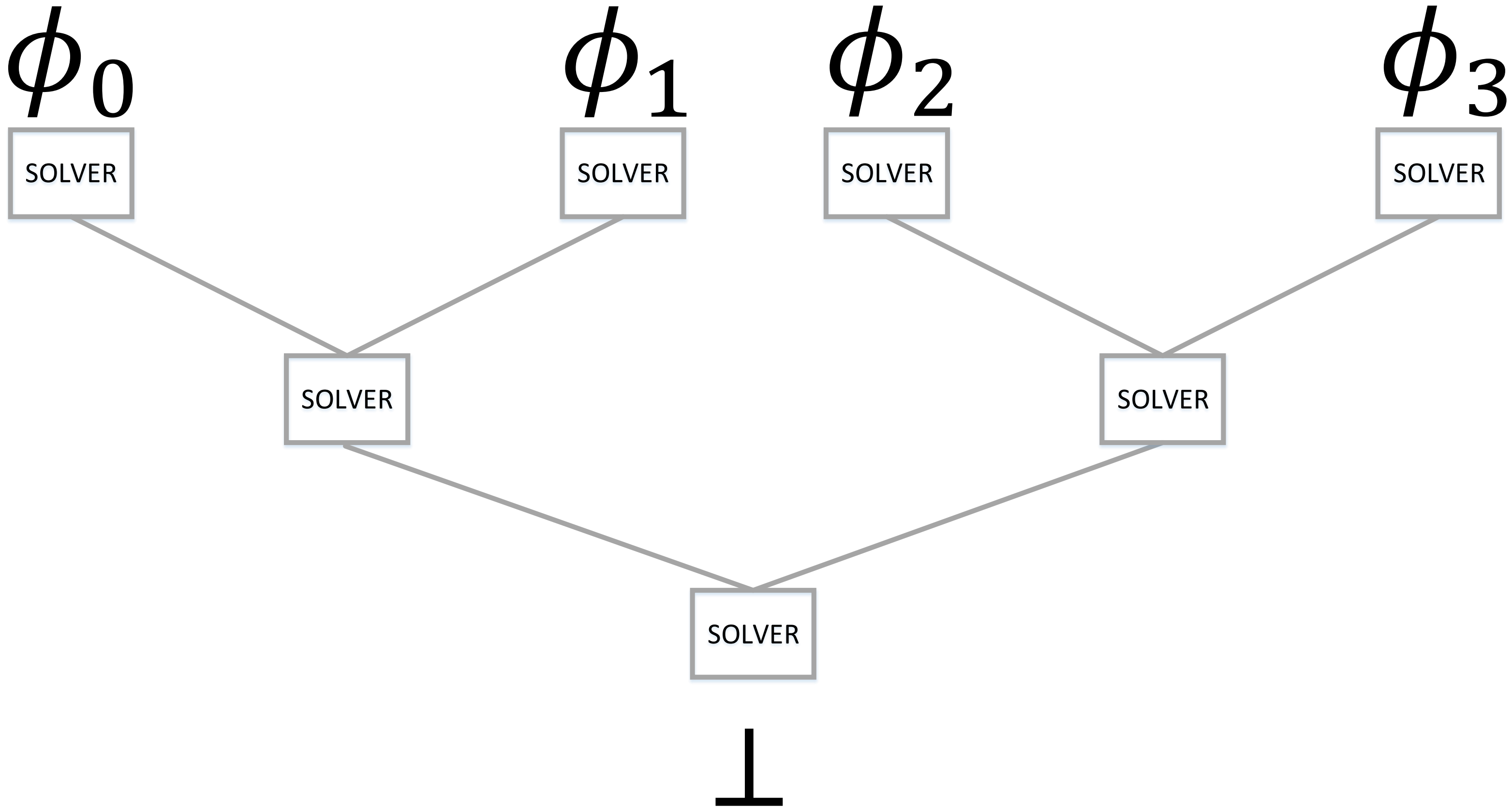
Control signals can depend on inputs that are independent from each other

$$\forall \vec{a} \;\; \exists \vec{c} \;\; \forall \vec{a}' \;\; \exists \vec{c}' \;\; \forall \vec{a}'' \;\; \exists \vec{c}'' \;\; \ldots \;\; . \Phi$$

- 1 level per ∀ ∃ - alternation
- $2^{|\vec{a}|}$ nodes per level

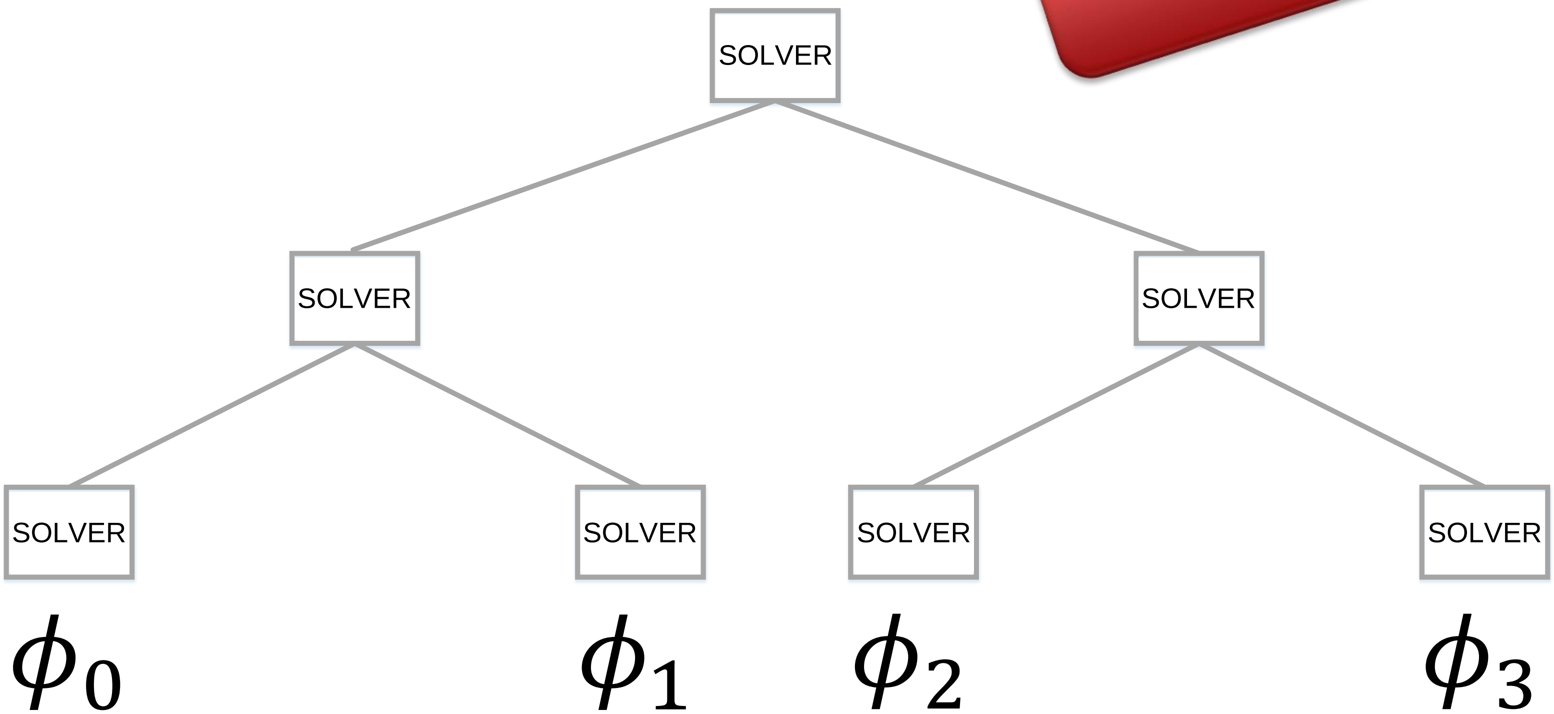
# Implementation

## Input Partitions $\phi_i$



# Implementation

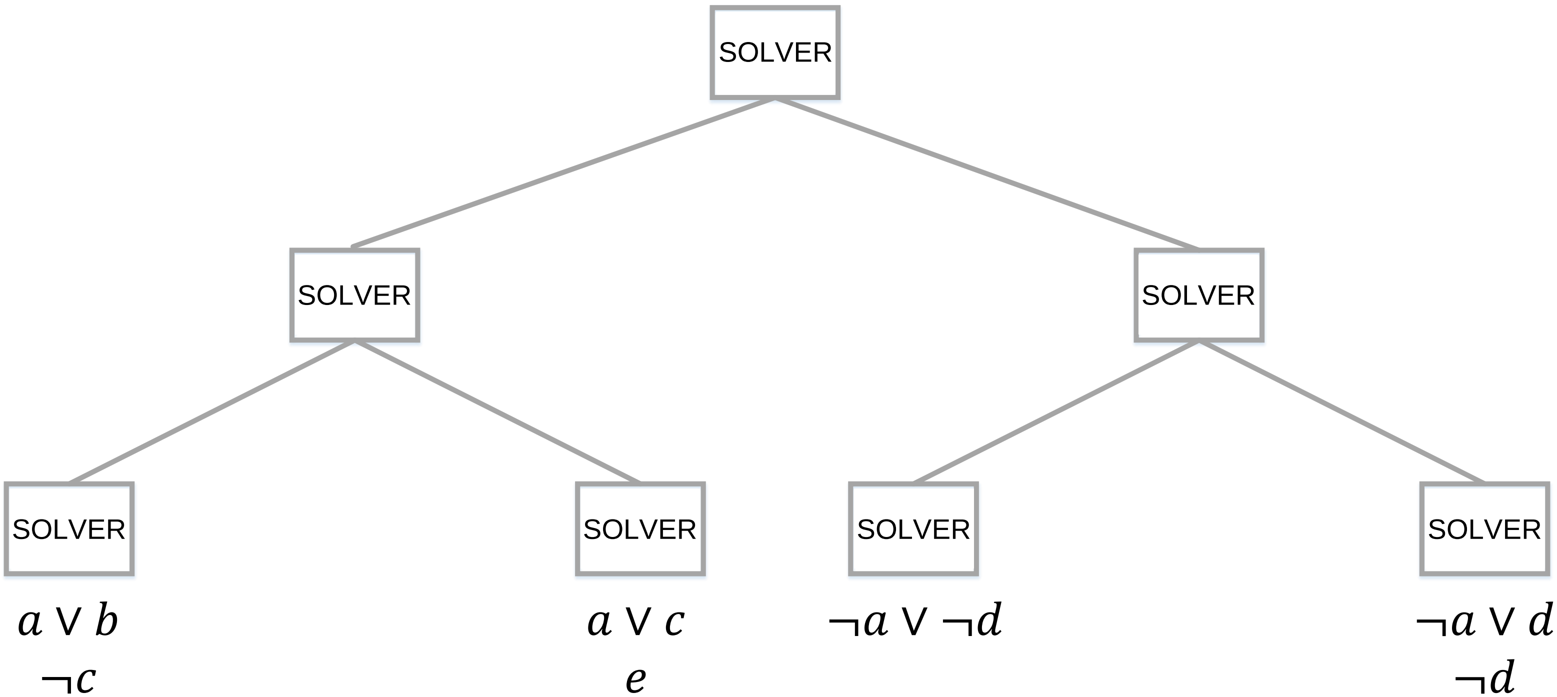
**Extension of modular SAT**
**Bayles et al.,(FMCAD 2013 )**




Input Partitions $\phi_i$

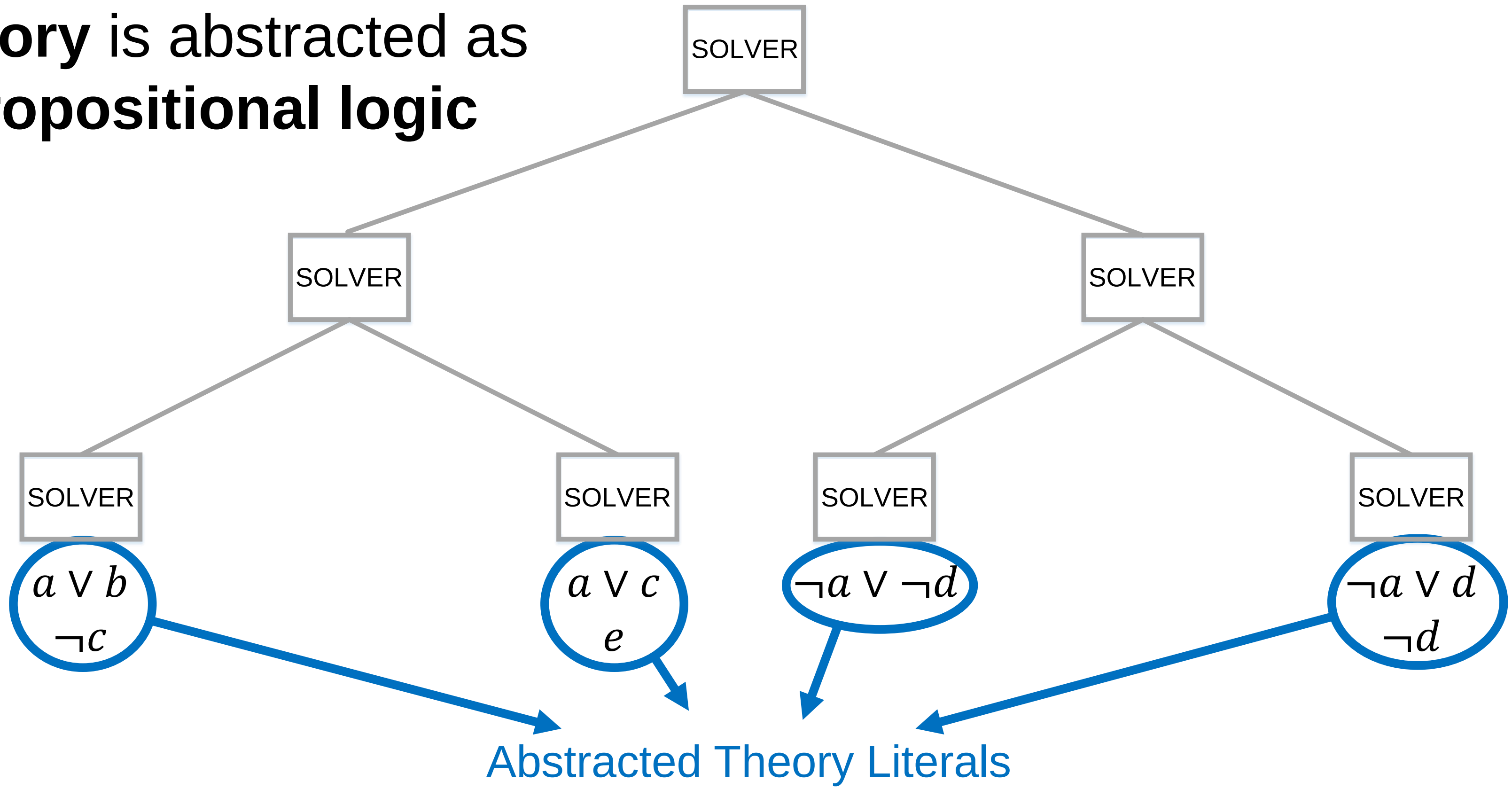
# Example

# Example

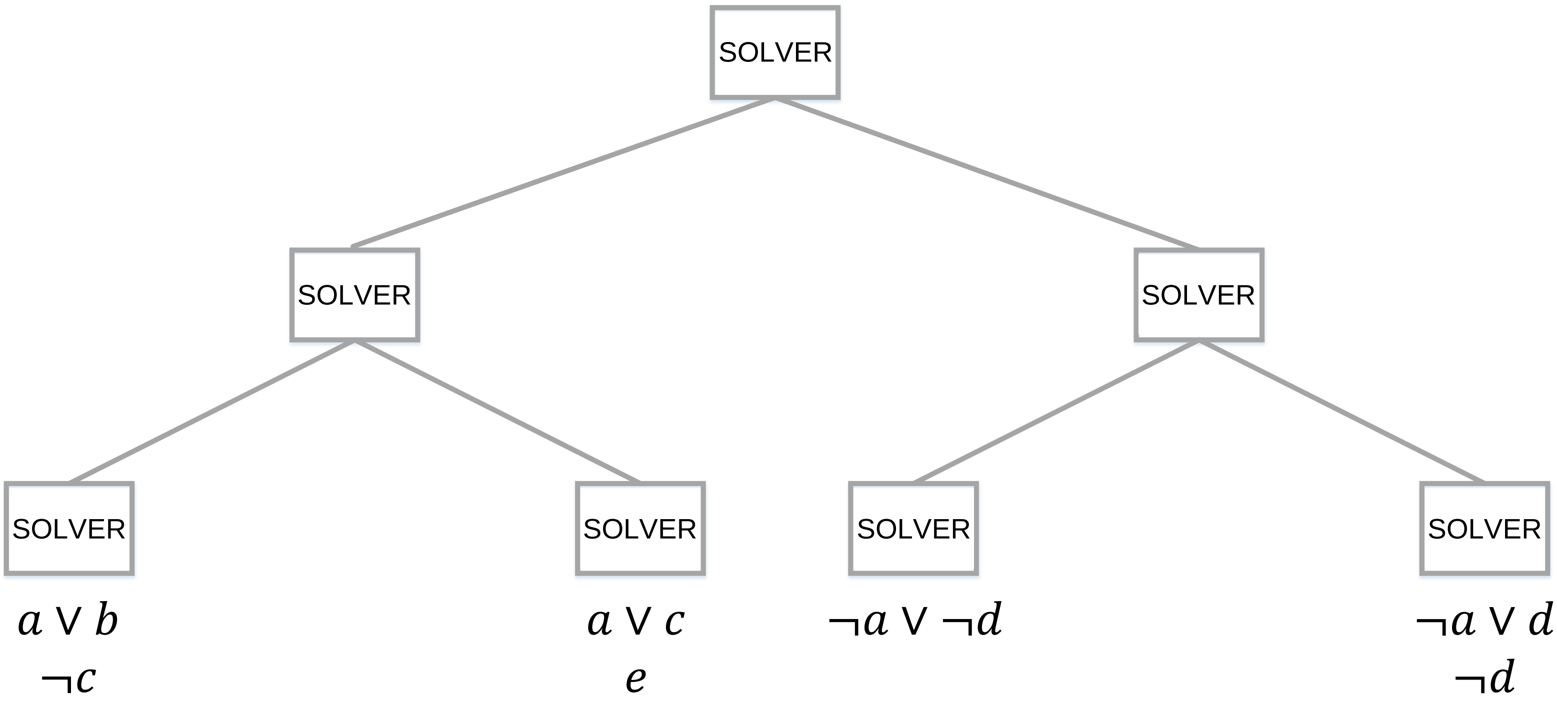

SOLVER
SOLVER
SOLVER
SOLVER
SOLVER
SOLVER
SOLVER
$a \lor b$
$\neg c$
$a \lor c$
$e$
$\neg a \lor \neg d$
$\neg a \lor d$
$\neg d$

# Example

**Theory** is abstracted as **propositional logic**

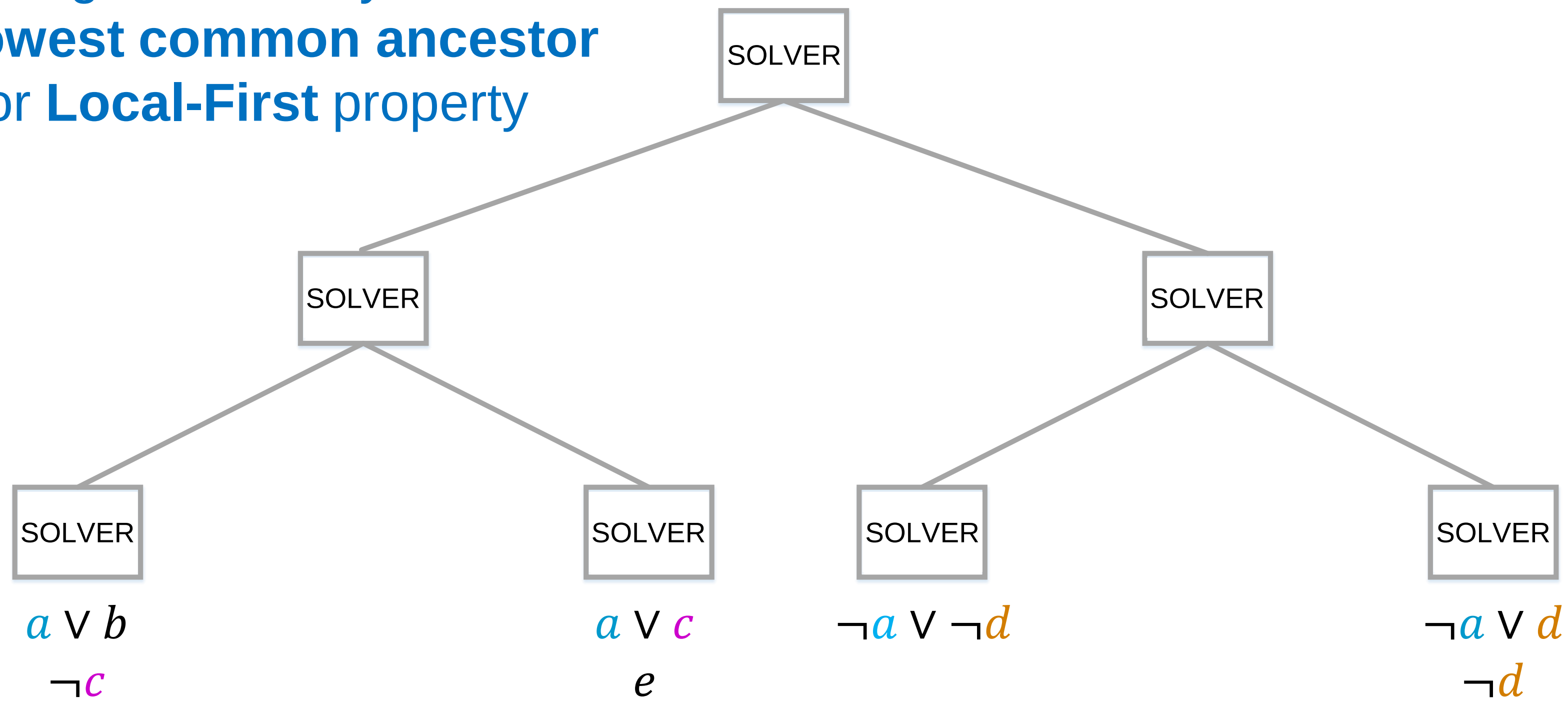


# Example


SOLVER
SOLVER
SOLVER
SOLVER
SOLVER
SOLVER
SOLVER
$a \lor b$
$\neg c$
$a \lor c$
$e$
$\neg a \lor \neg d$
$\neg a \lor d$
$\neg d$

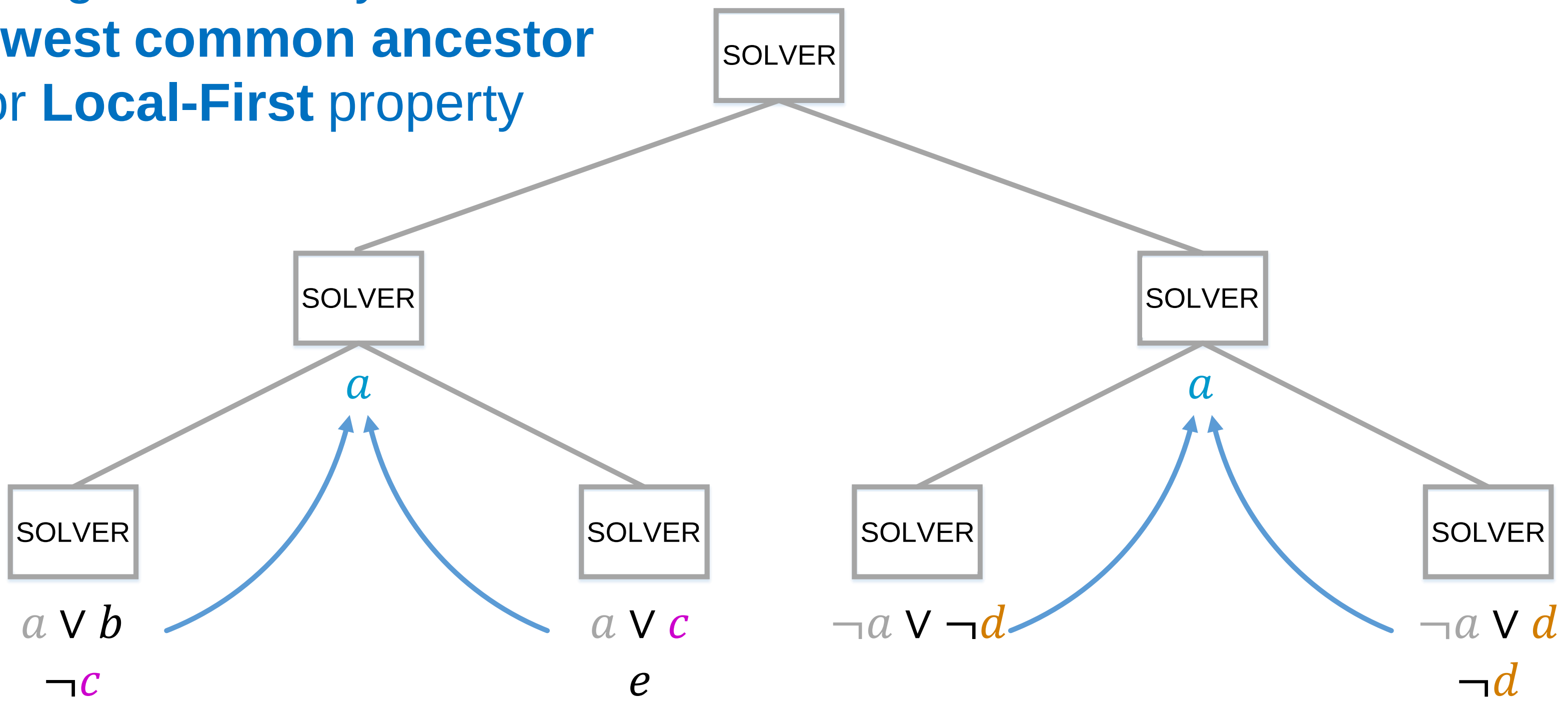
# Example

Assign **Global** Symbols
to **lowest common ancestor**
for **Local-First** property



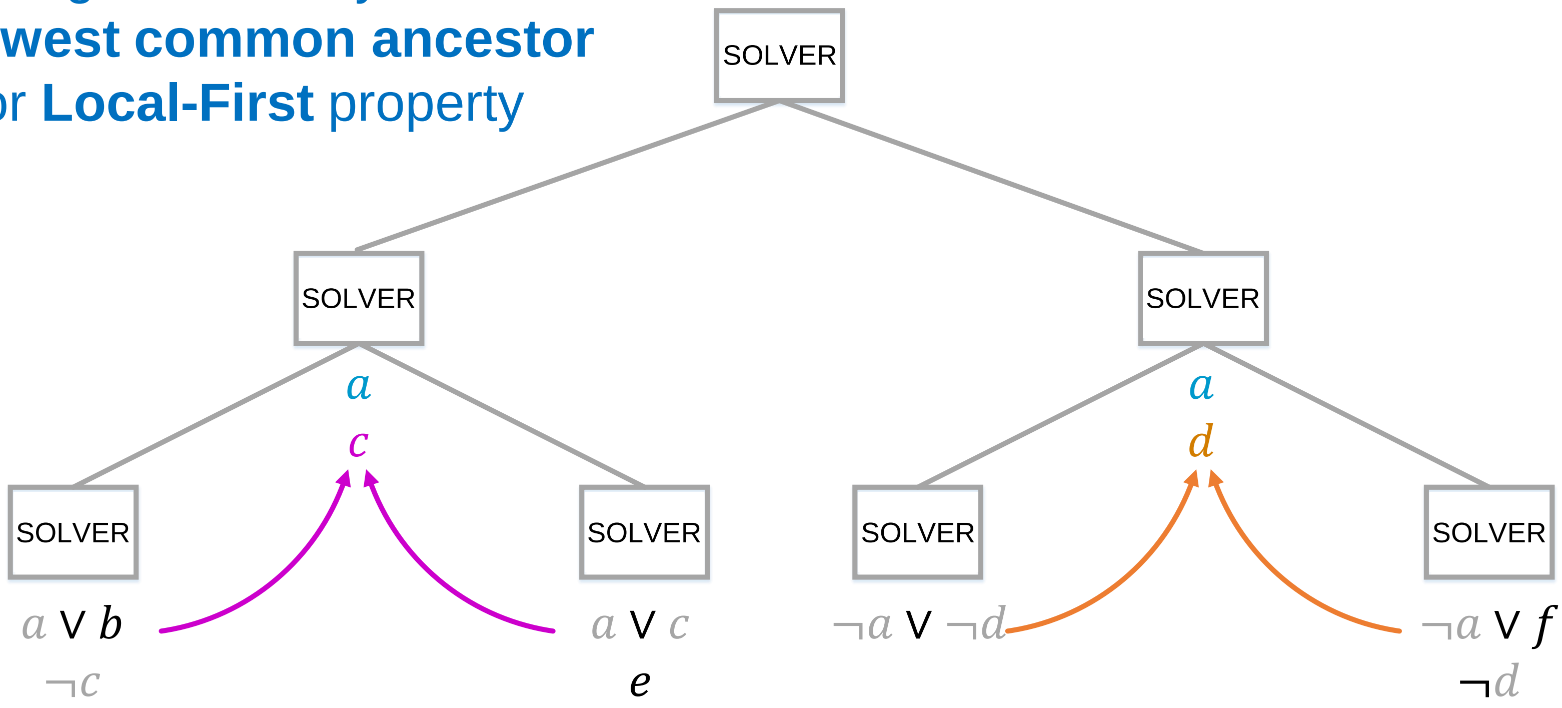
# Example

Assign **Global** Symbols
to **lowest common ancestor**
for **Local-First** property



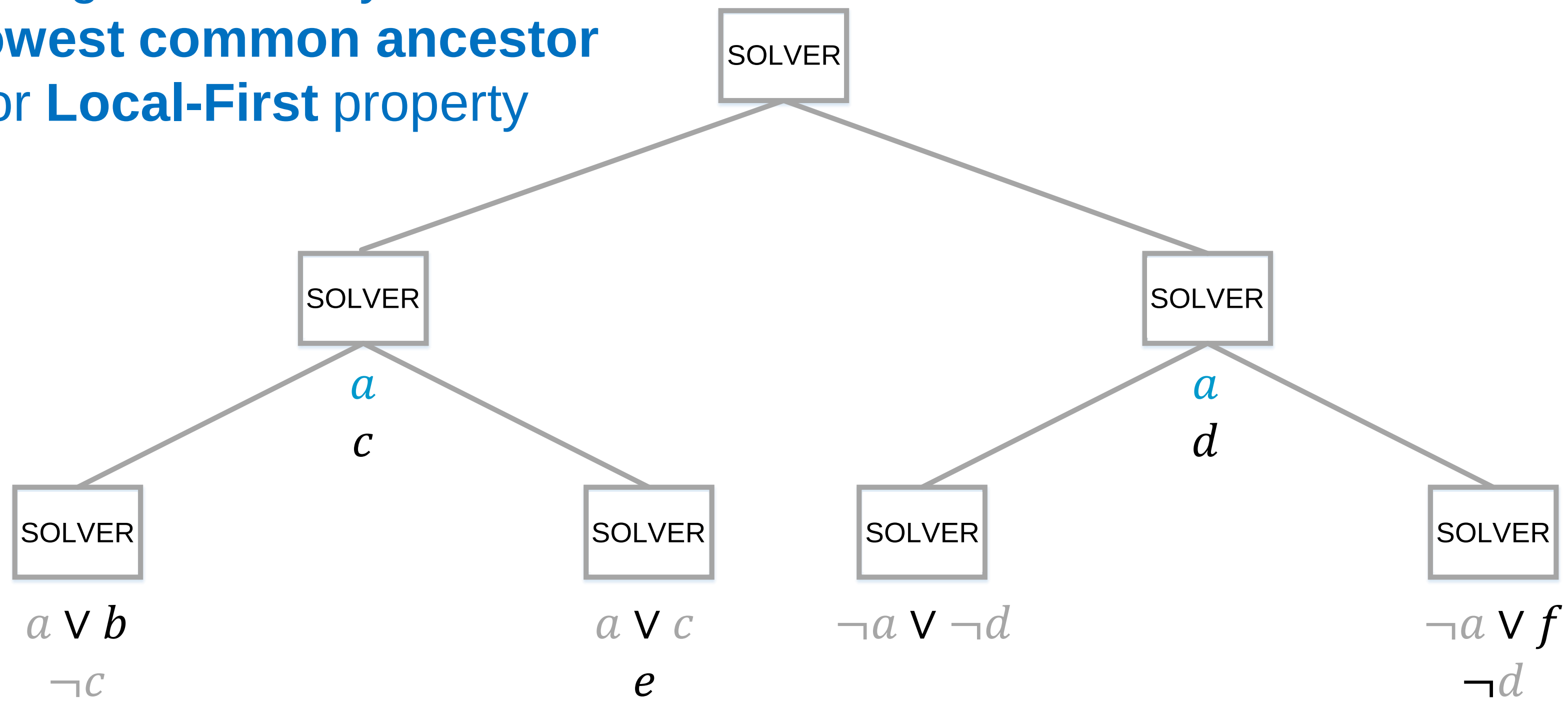
# Example

Assign **Global** Symbols
to **lowest common ancestor**
for **Local-First** property



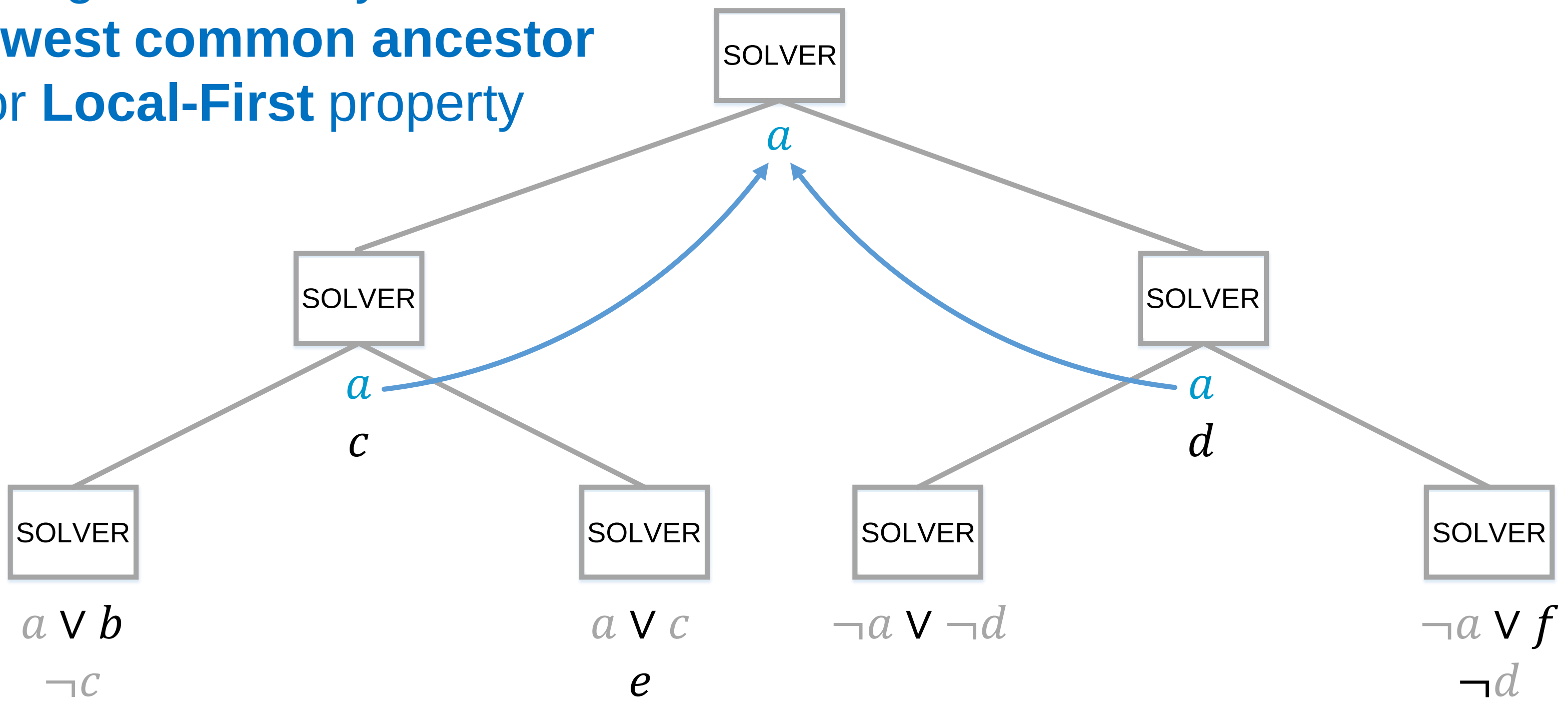
# Example

Assign **Global** Symbols
to **lowest common ancestor**
for **Local-First** property

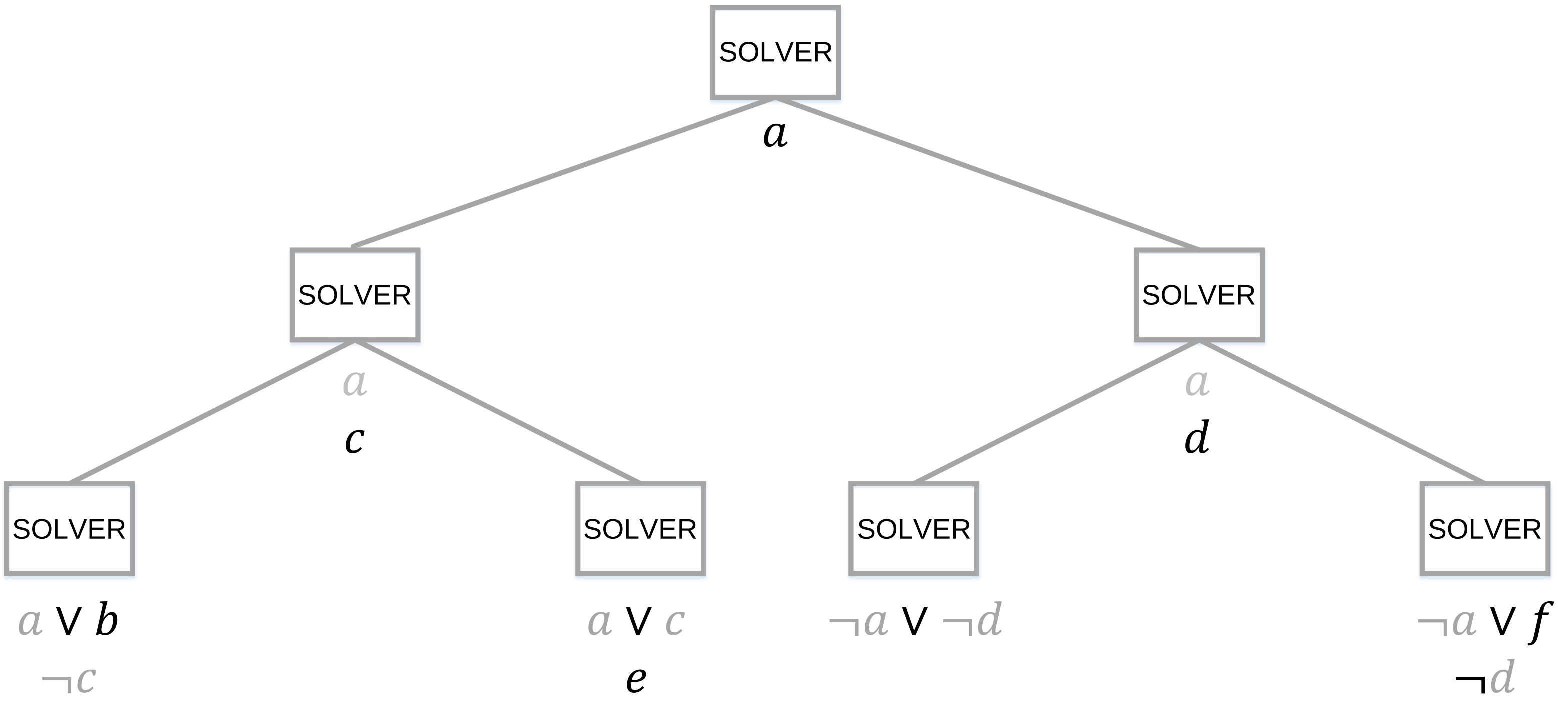


# Example

Assign **Global** Symbols
to **lowest common ancestor**
for **Local-First** property

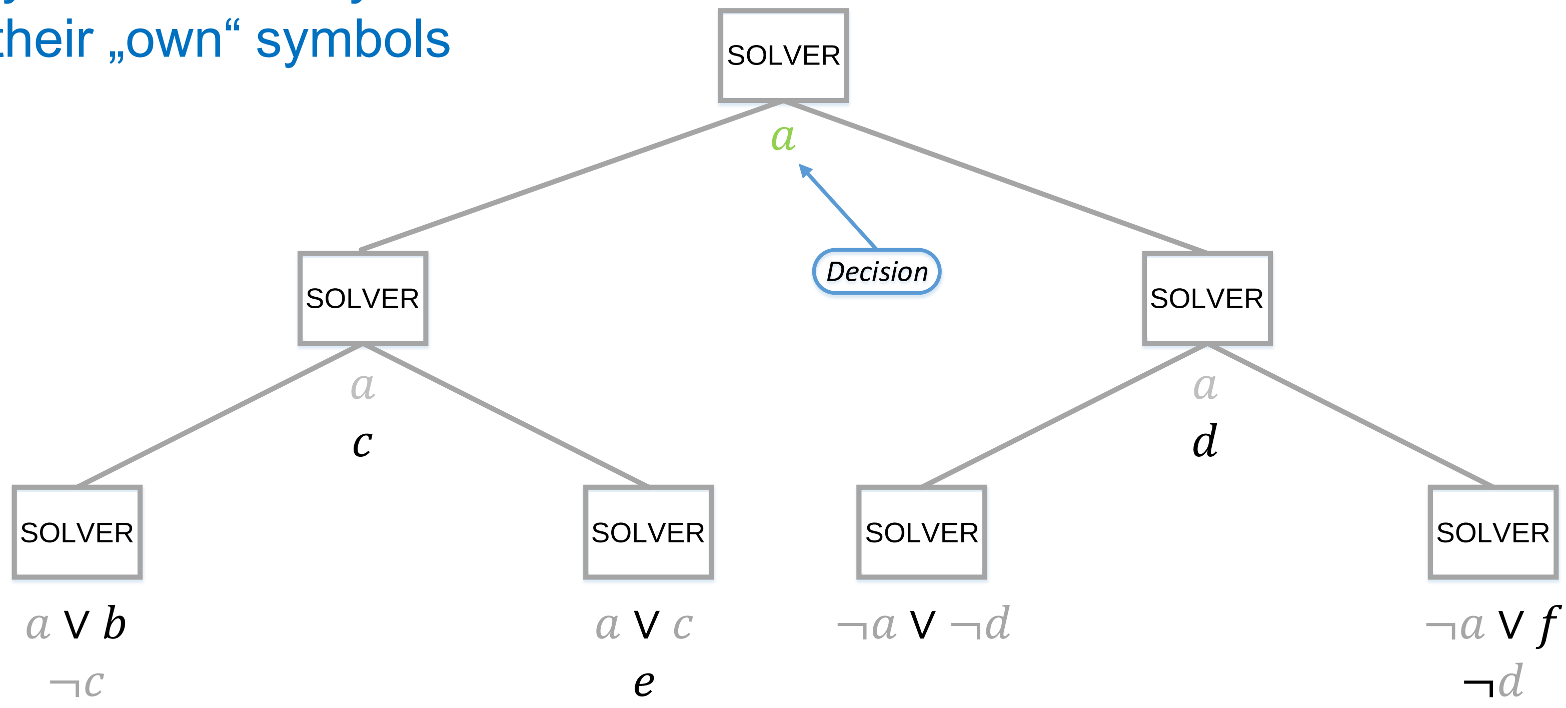


# Example


SOLVER
$a$
SOLVER
$a$
$c$
SOLVER
$a$
$d$
SOLVER
$a \lor b$
$\neg c$
SOLVER
$a \lor c$
$e$
SOLVER
$\neg a \lor \neg d$
SOLVER
$\neg a \lor f$
$\neg d$

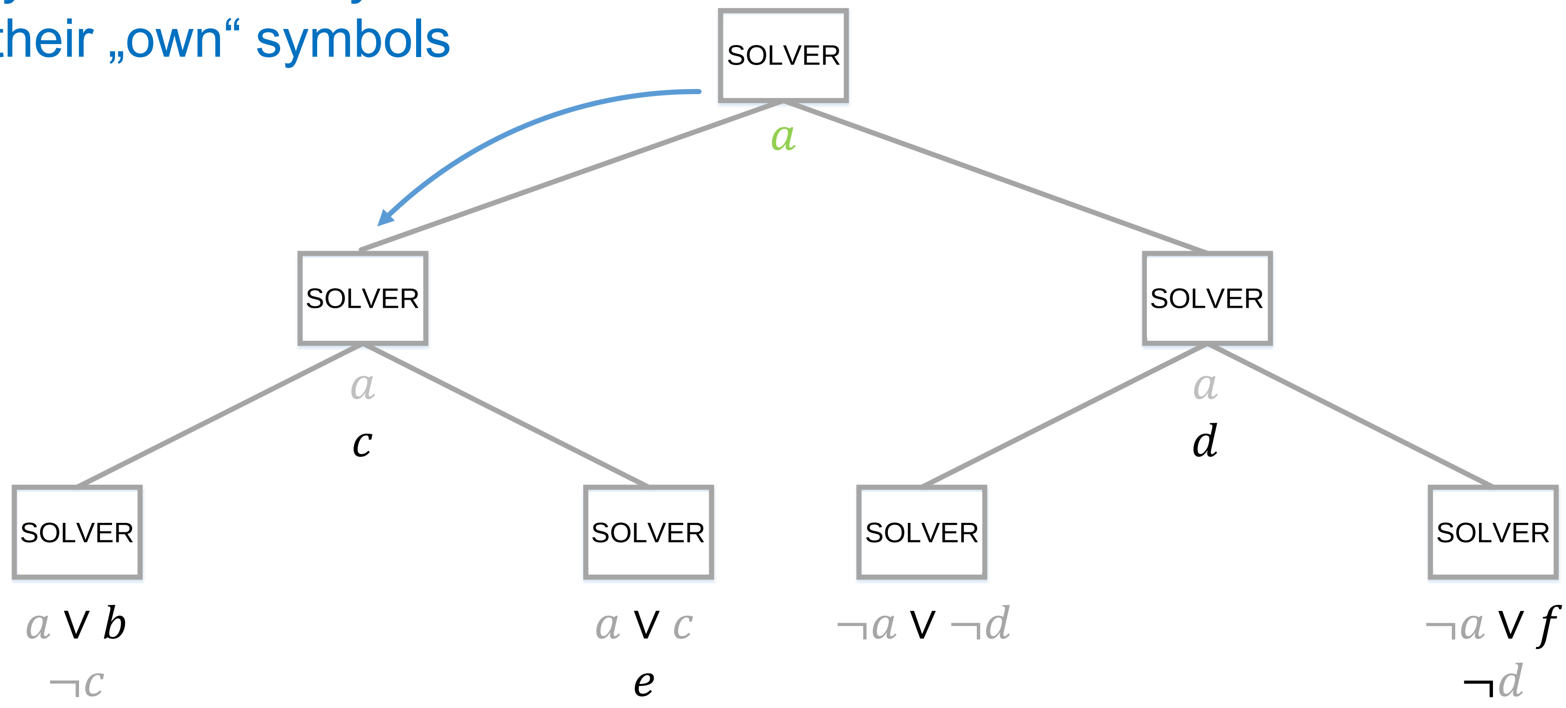
# Example

Every node can only decide their „own“ symbols



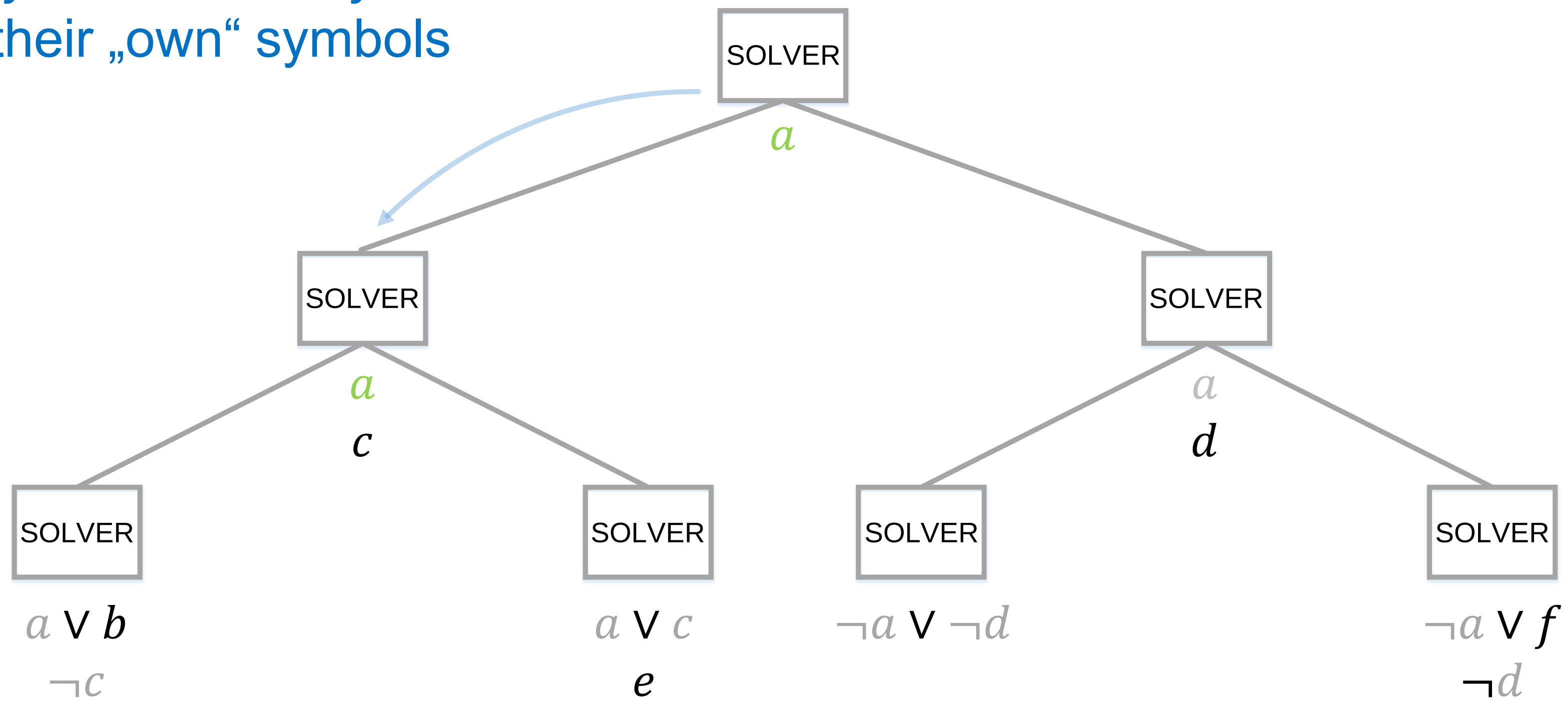
# Example

Every node can only decide their „own“ symbols



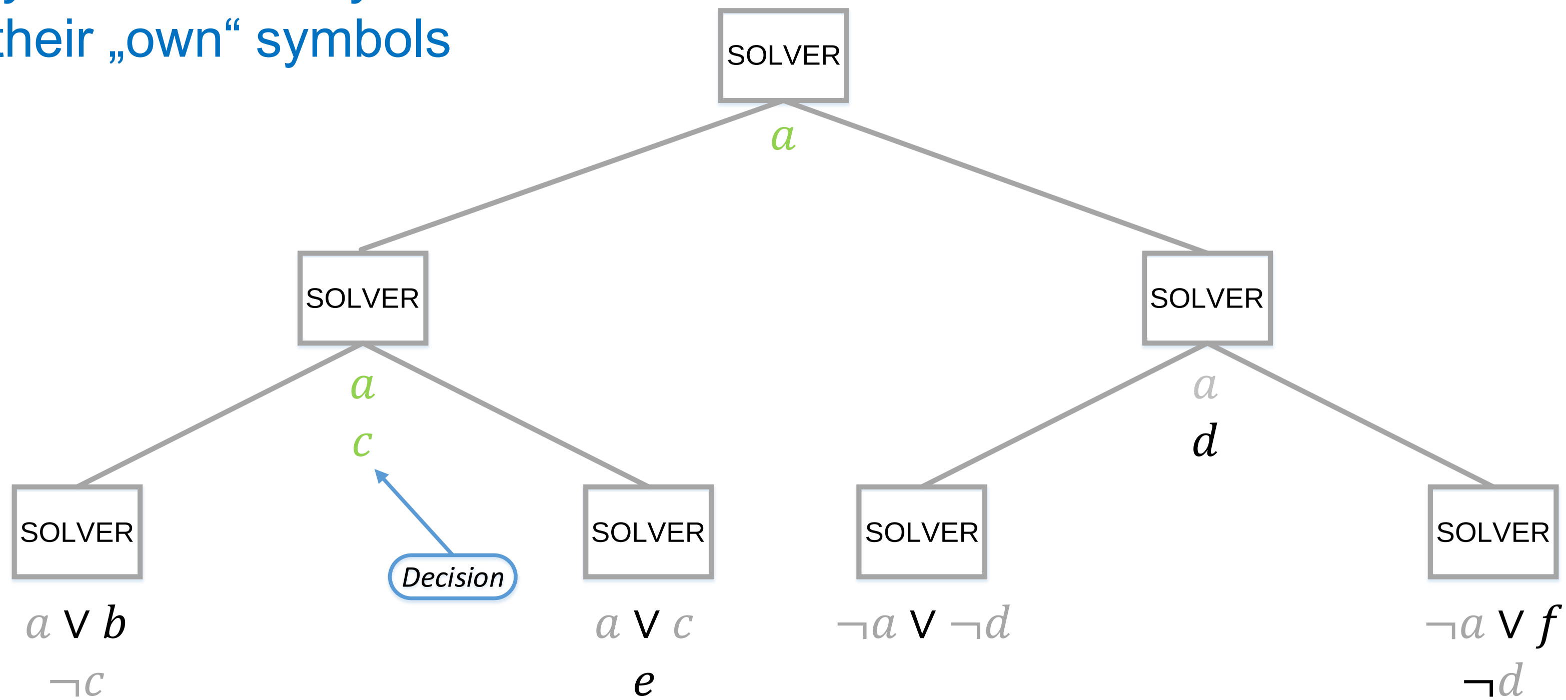
# Example

Every node can only decide their „own“ symbols



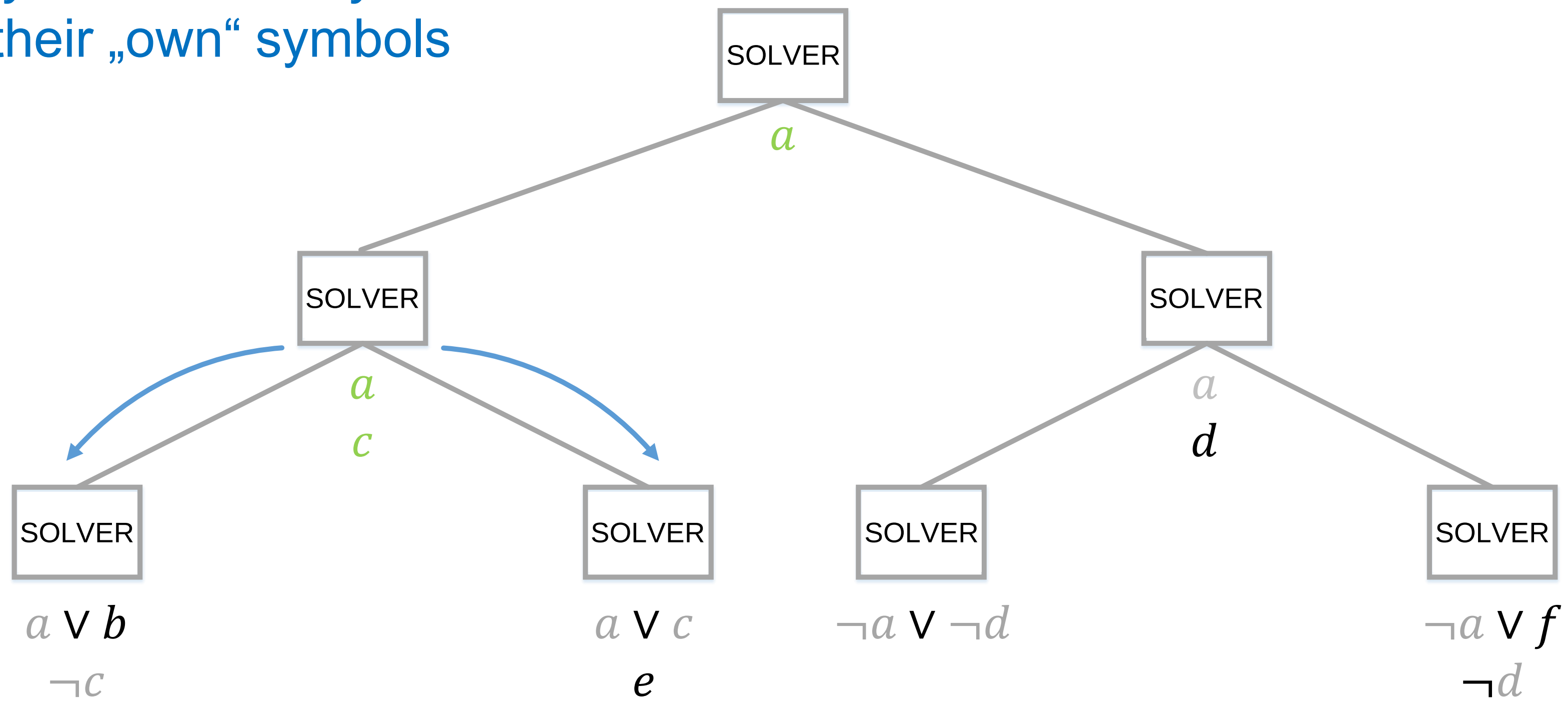
# Example

Every node can only decide their „own“ symbols



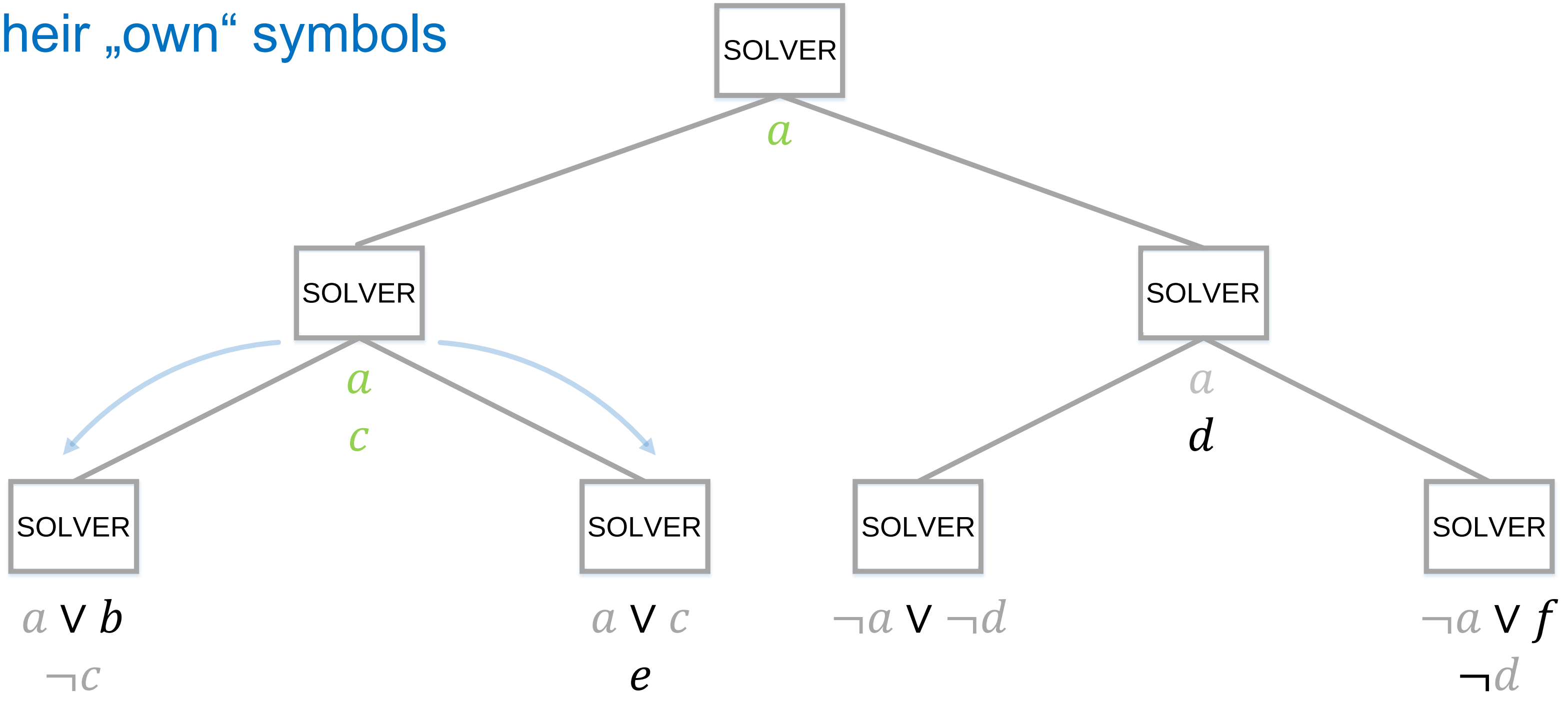
# Example

Every node can only decide their „own“ symbols



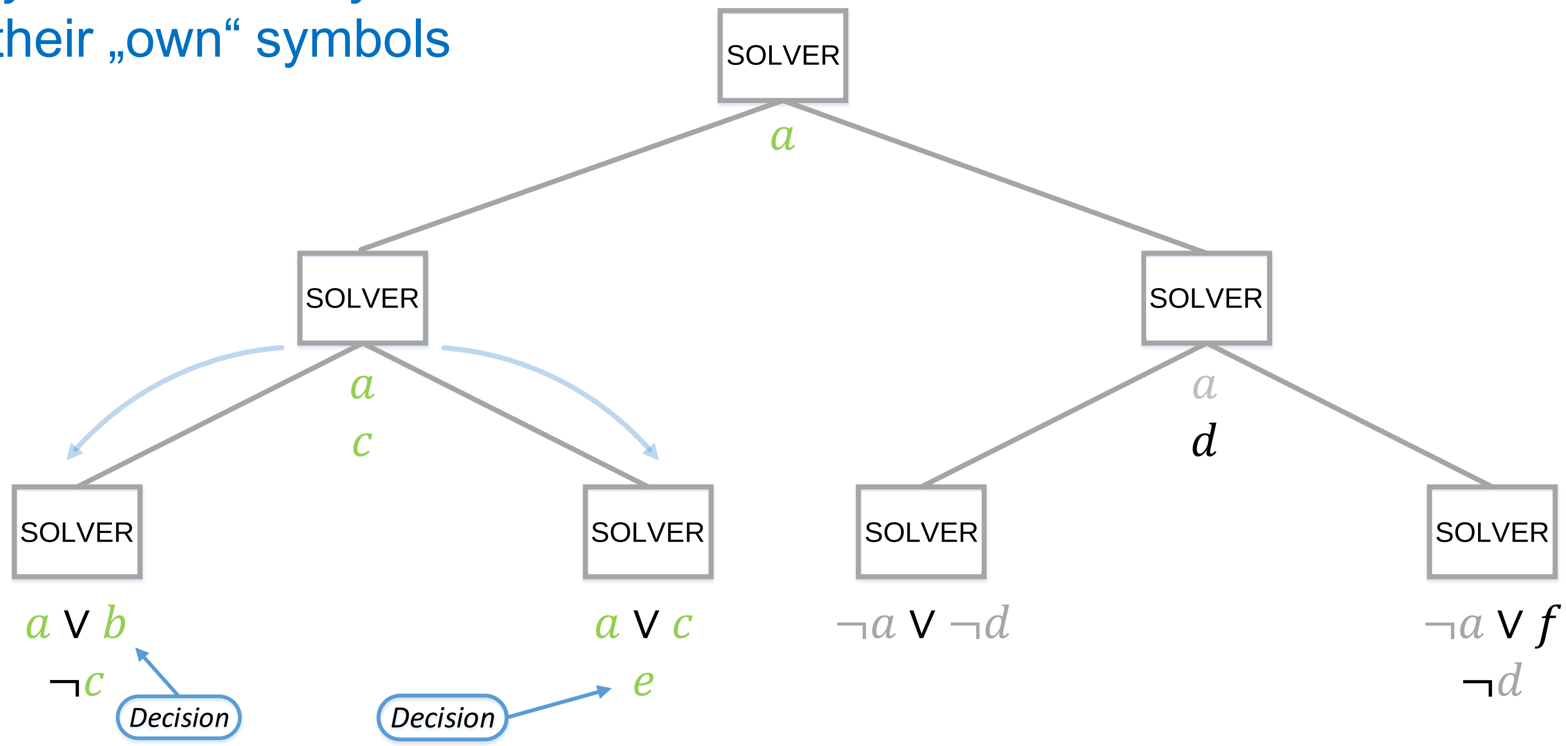
# Example

Every node can only decide their „own“ symbols

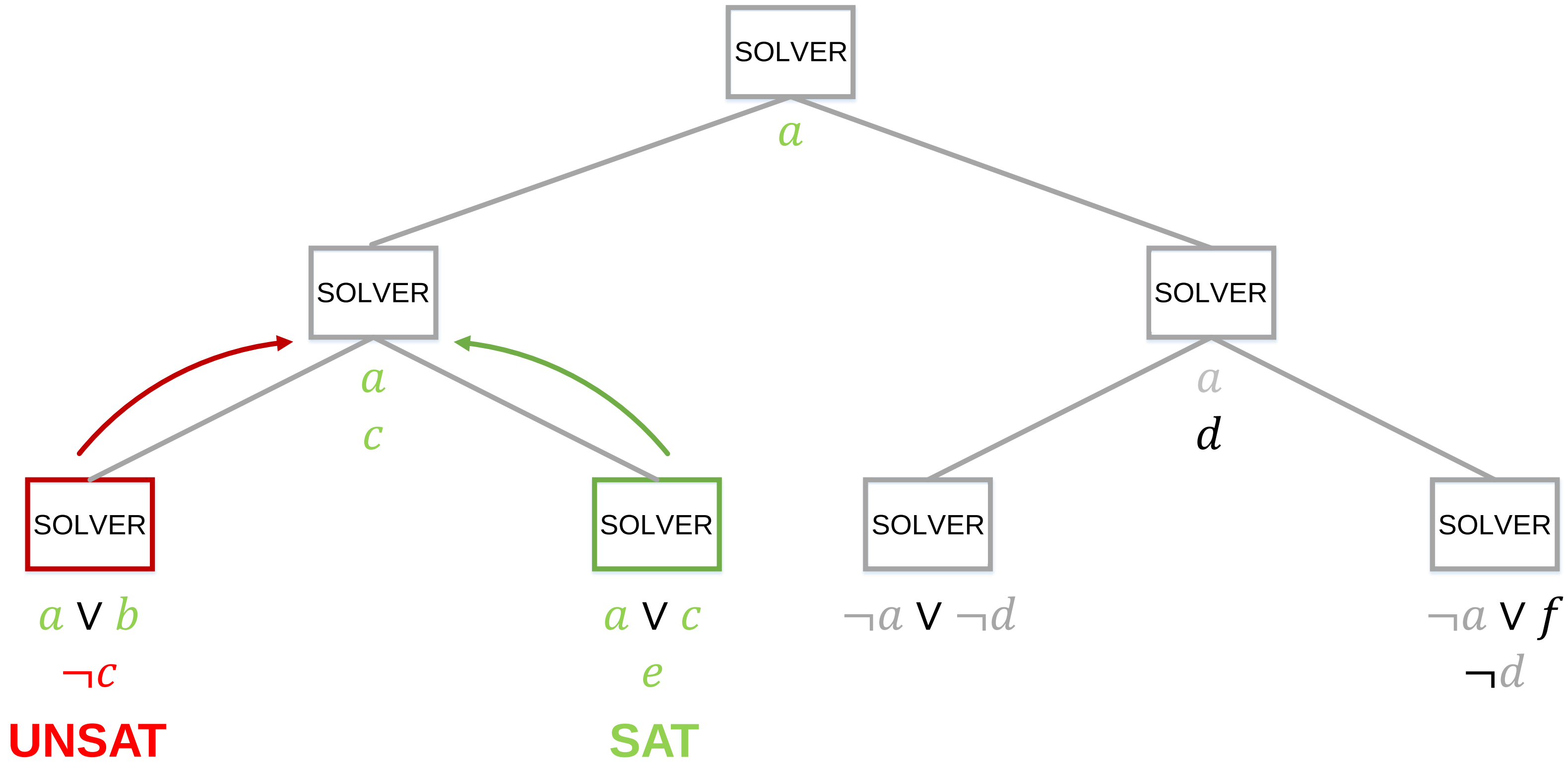


# Example

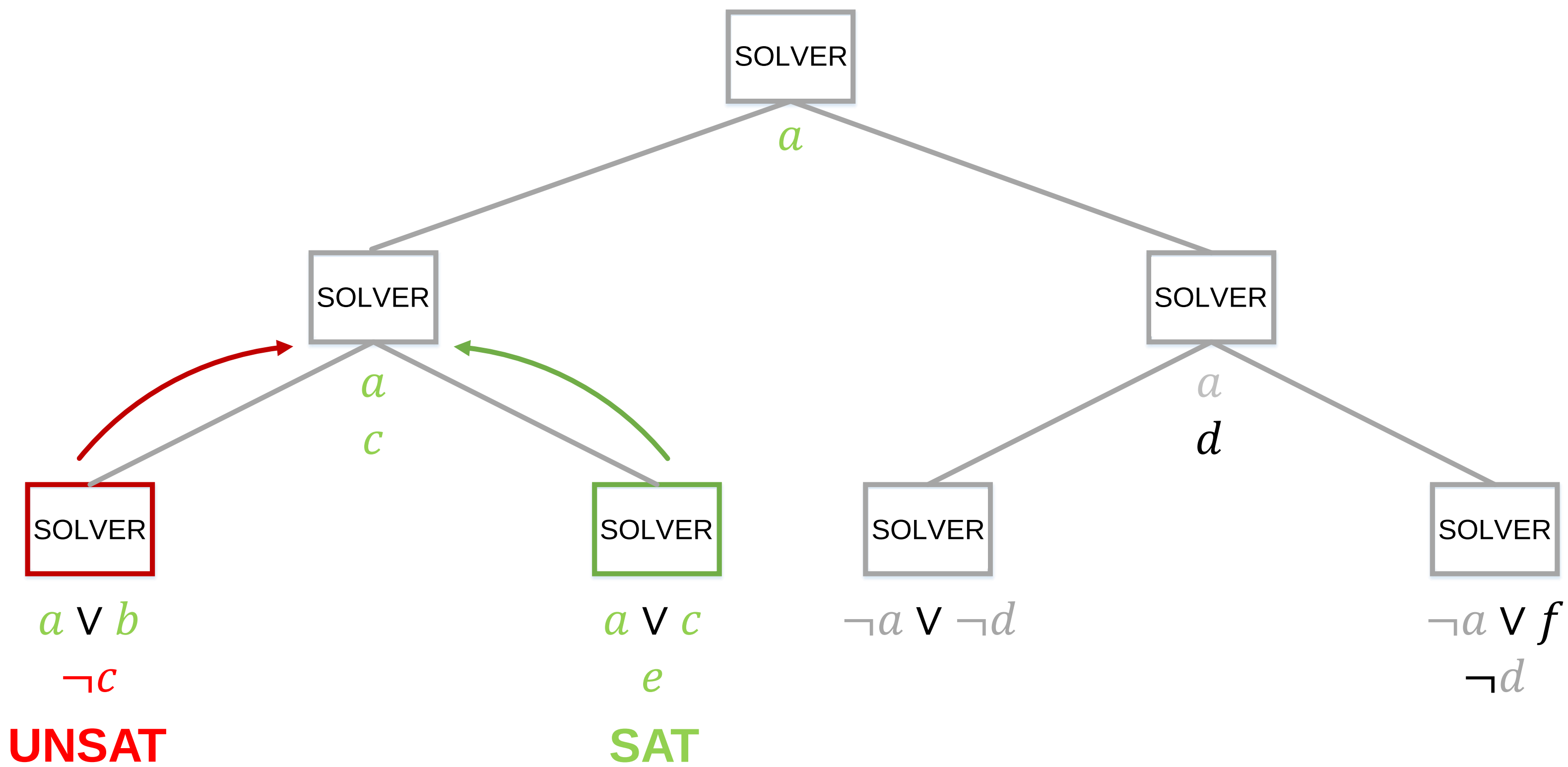
Every node can only decide their „own“ symbols



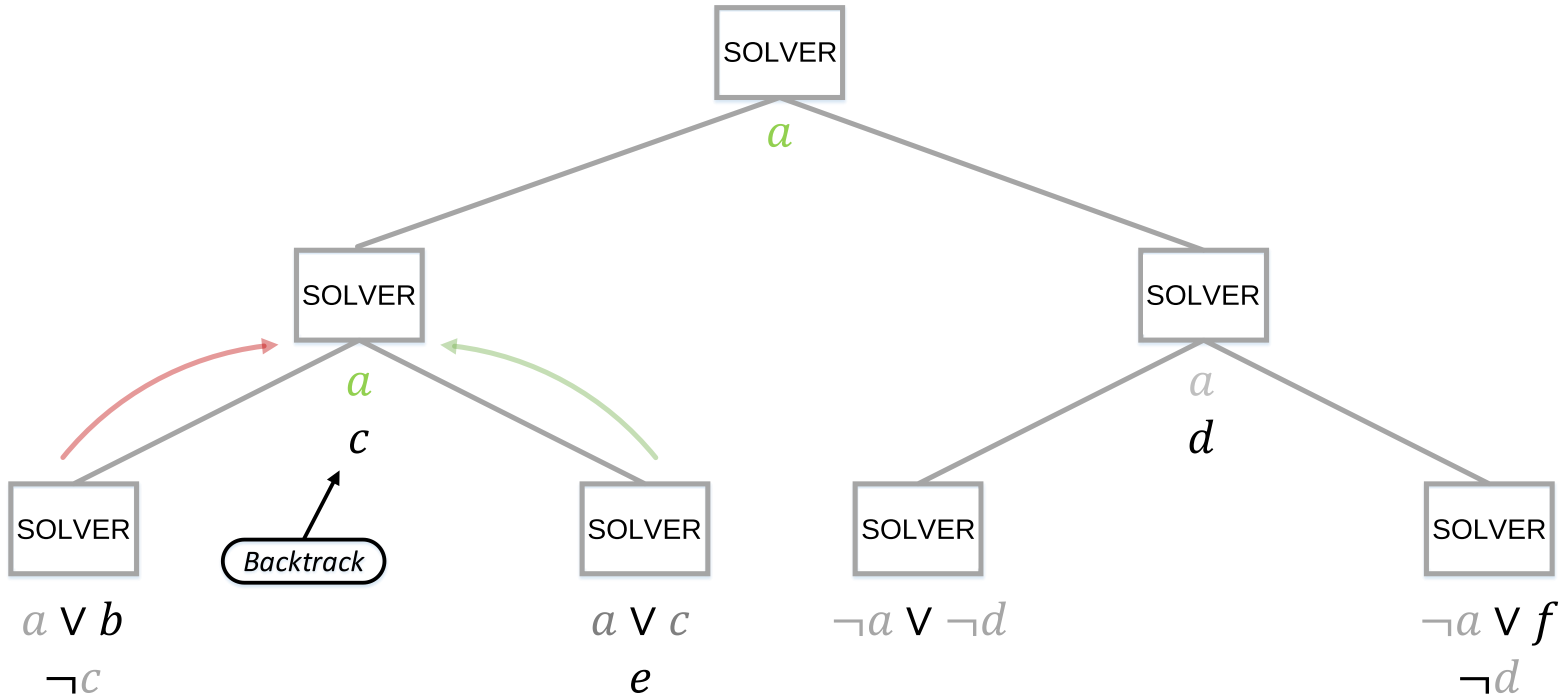
# Example


SOLVER
a
SOLVER
a
c
SOLVER
a
d
SOLVER
a ∨ b
¬c
UNSAT
SOLVER
a ∨ c
e
SAT
SOLVER
¬a ∨ ¬d
SOLVER
¬a ∨ f
¬d

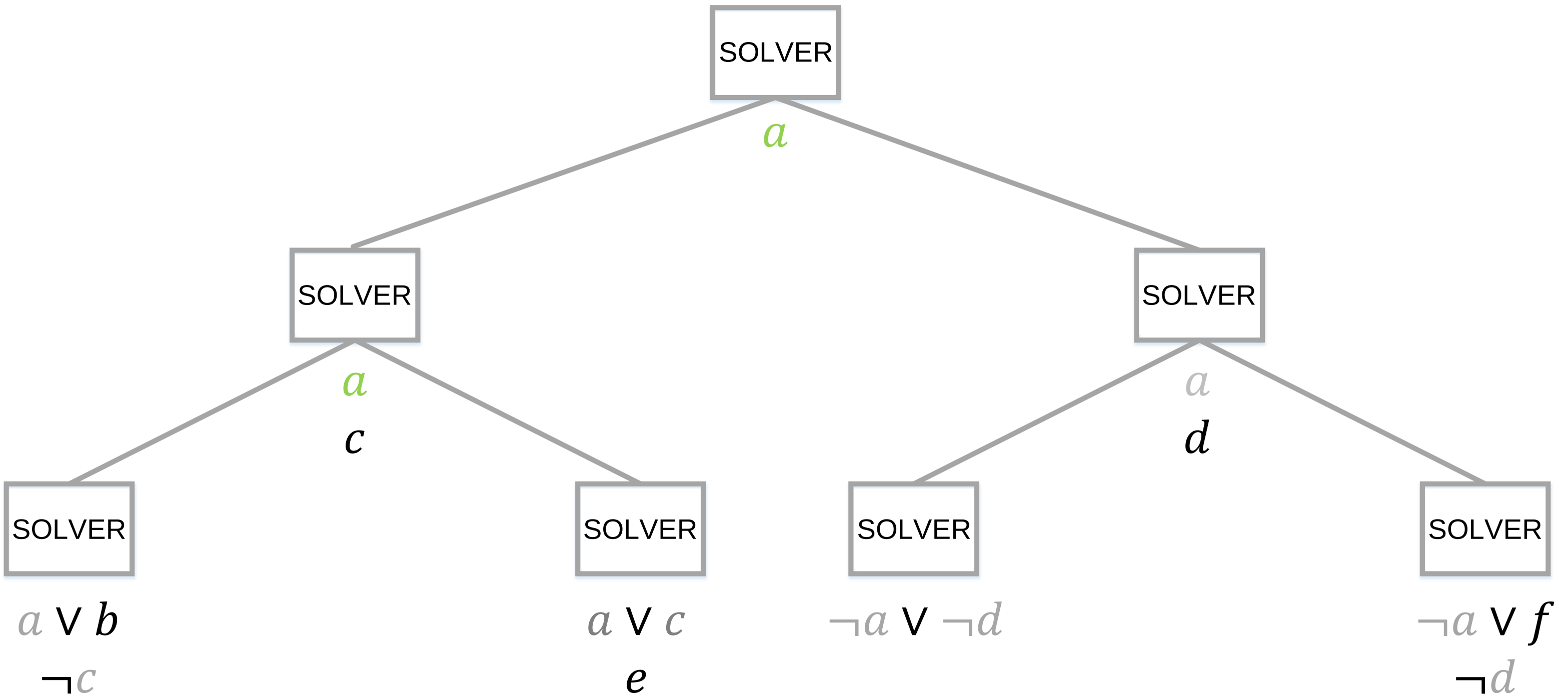
# Example


SOLVER
$a$
SOLVER
SOLVER
$a$
$c$
$a$
$d$
SOLVER
SOLVER
SOLVER
SOLVER
$a \vee b$
$\neg c$
UNSAT
$a \vee c$
$e$
SAT
$\neg a \vee \neg d$
$\neg a \vee f$
$\neg d$

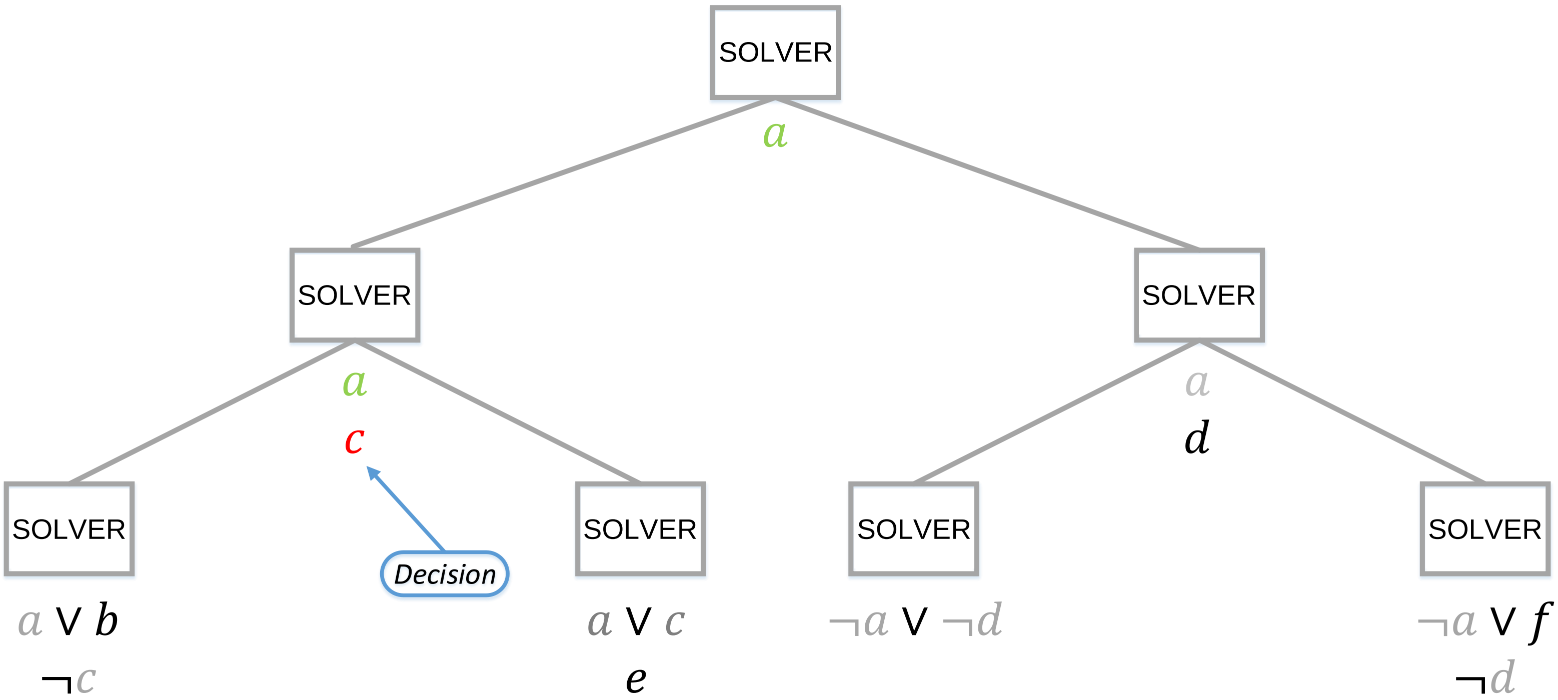
# Example


SOLVER
$a$
SOLVER
SOLVER
$a$
$c$
$a$
$d$
SOLVER
Backtrack
SOLVER
SOLVER
SOLVER
$a \vee b$
$\neg c$
$a \vee c$
$e$
$\neg a \vee \neg d$
$\neg a \vee f$
$\neg d$

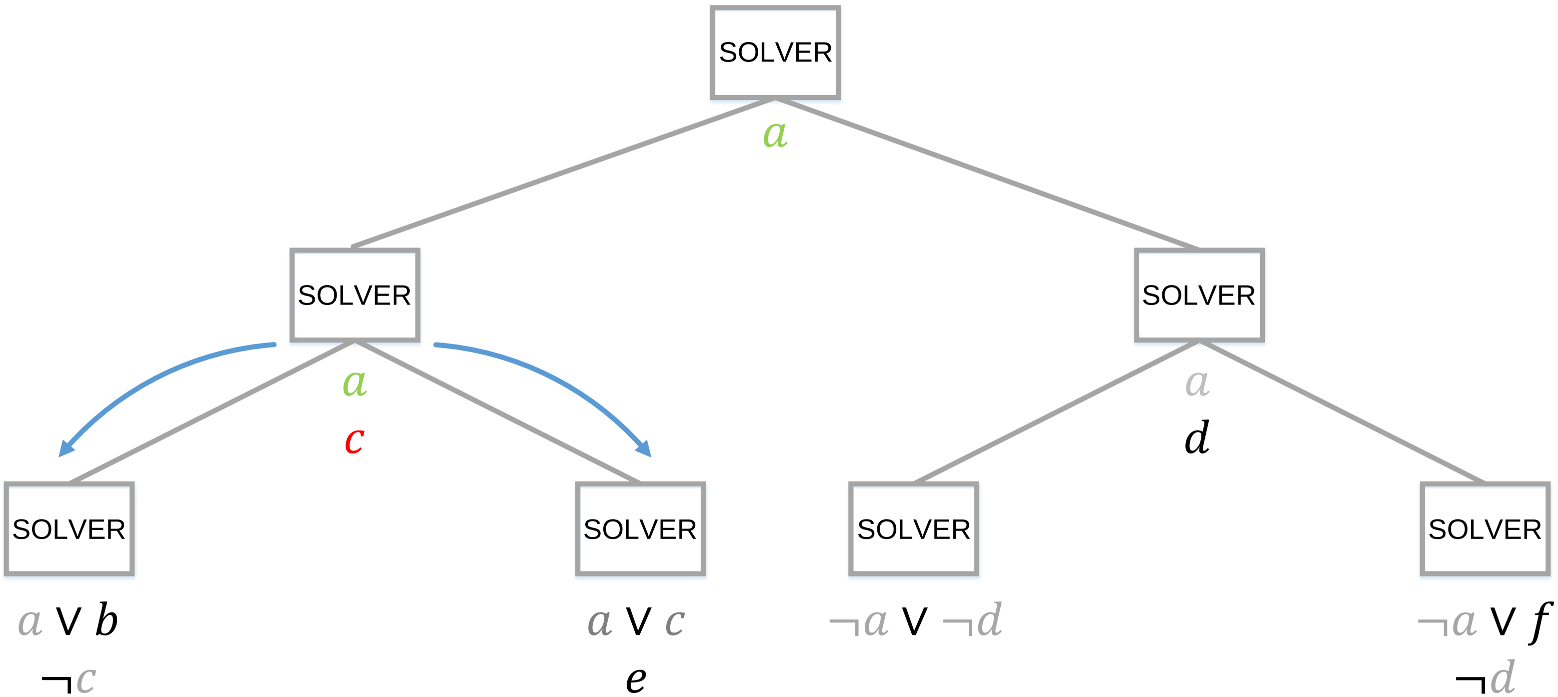
# Example


SOLVER
$a$
SOLVER
$a$
$c$
SOLVER
$a$
$d$
SOLVER
$a \lor b$
$\neg c$
SOLVER
$a \lor c$
$e$
SOLVER
$\neg a \lor \neg d$
SOLVER
$\neg a \lor f$
$\neg d$

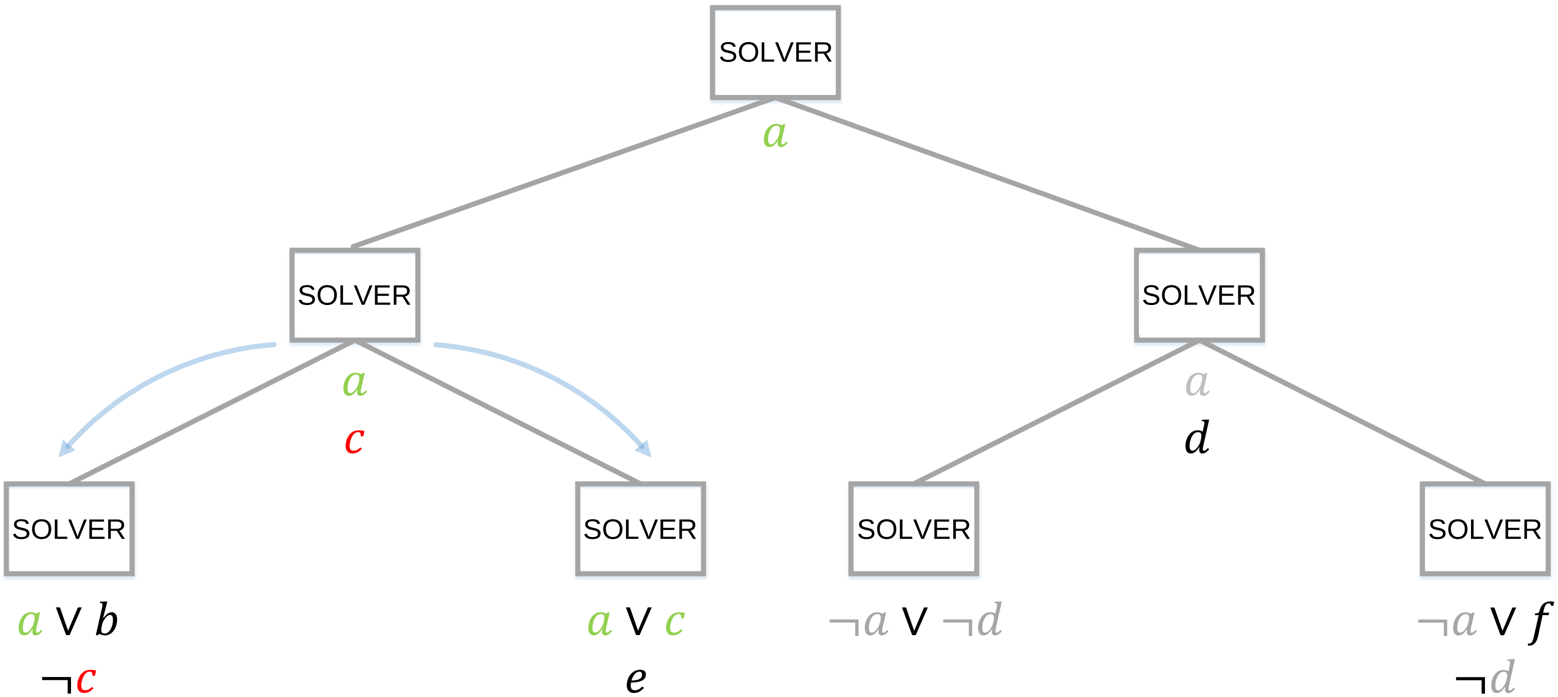
# Example


SOLVER
a
SOLVER
SOLVER
a
c
a
d
SOLVER
Decision
SOLVER
SOLVER
SOLVER
a ∨ b
¬c
a ∨ c
e
¬a ∨ ¬d
¬a ∨ f
¬d

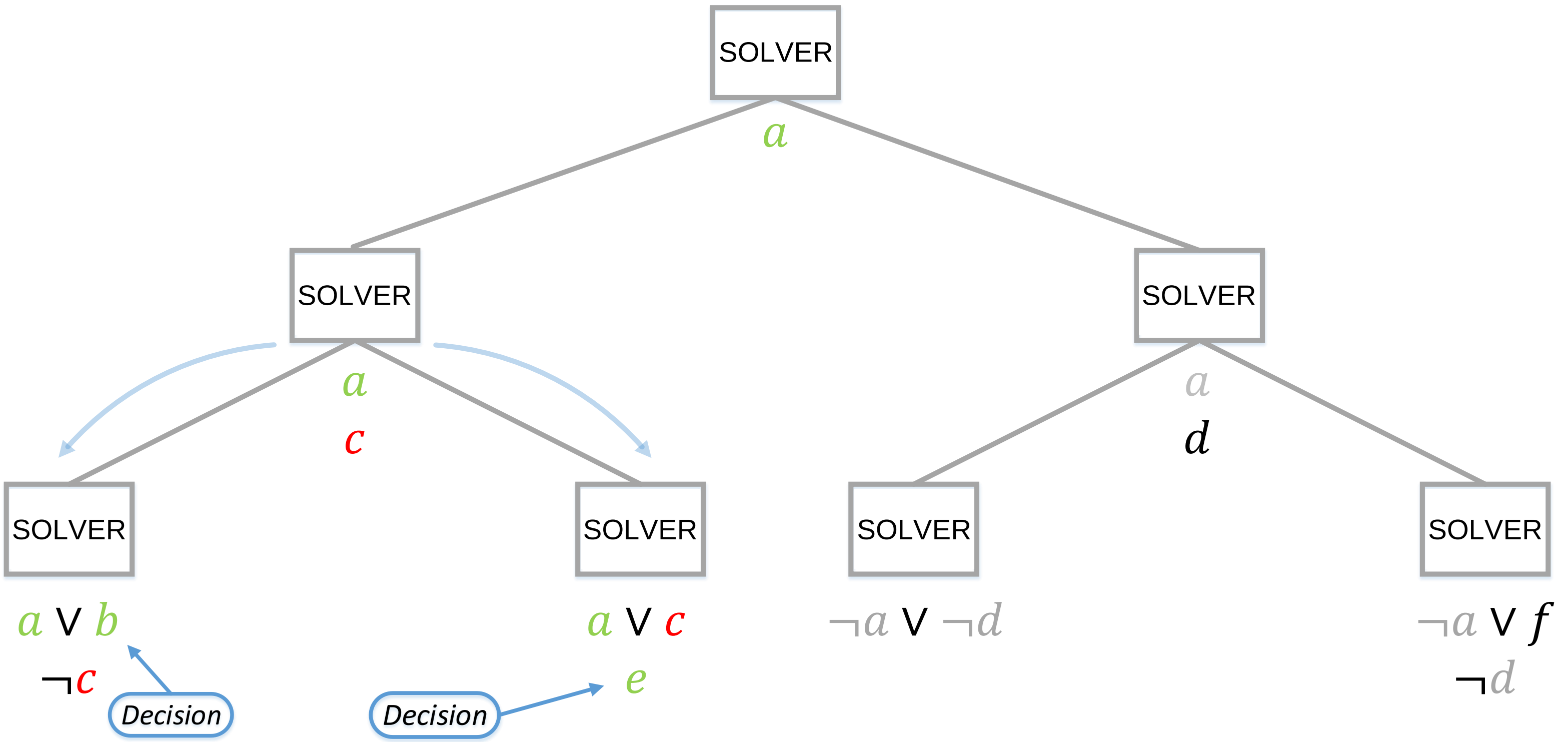
# Example


SOLVER
$a$
SOLVER
SOLVER
$a$
$c$
$a$
$d$
SOLVER
SOLVER
SOLVER
SOLVER
$a \vee b$
$\neg c$
$a \vee c$
$e$
$\neg a \vee \neg d$
$\neg a \vee f$
$\neg d$

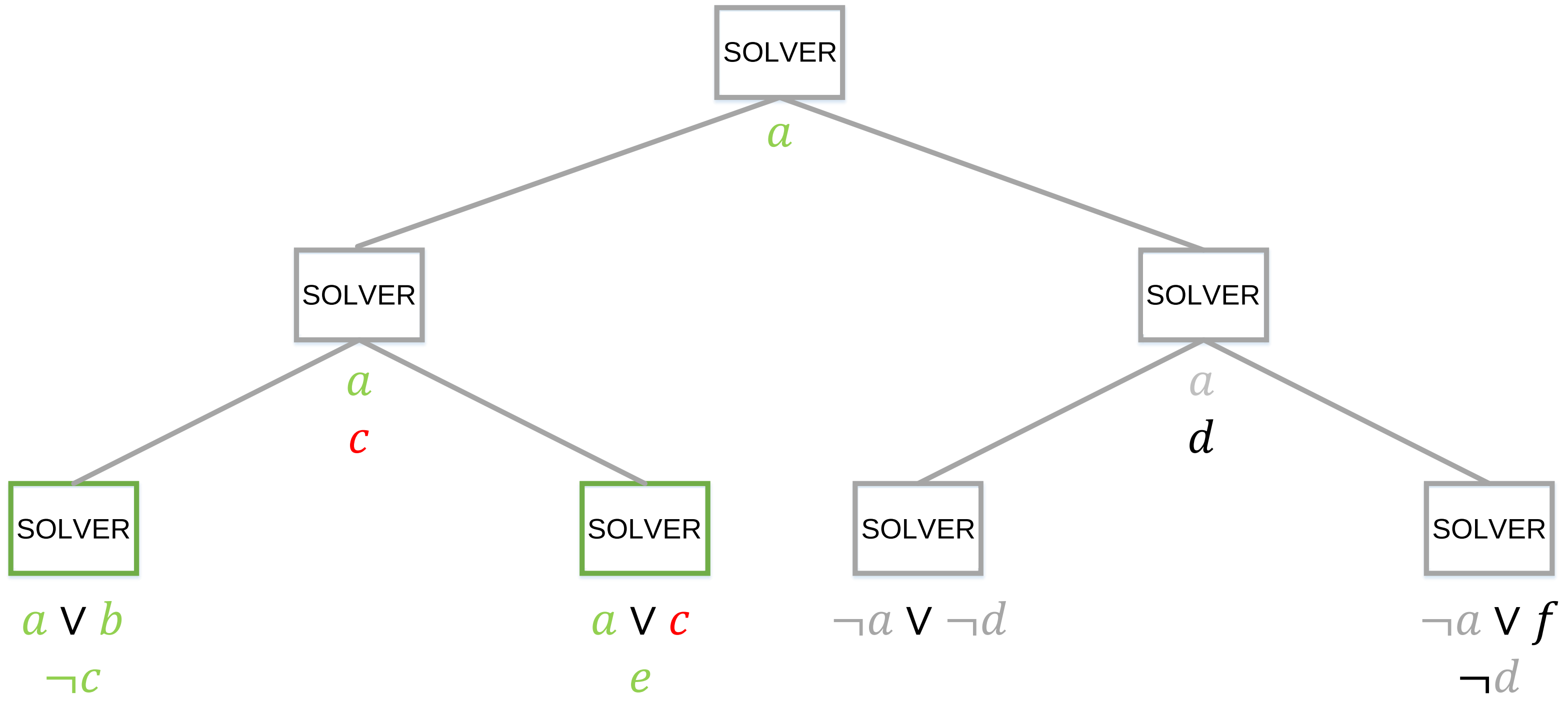
# Example


SOLVER
$a$
SOLVER
SOLVER
$a$
$c$
$a$
$d$
SOLVER
SOLVER
SOLVER
SOLVER
$a \vee b$
$\neg c$
$a \vee c$
$e$
$\neg a \vee \neg d$
$\neg a \vee f$
$\neg d$

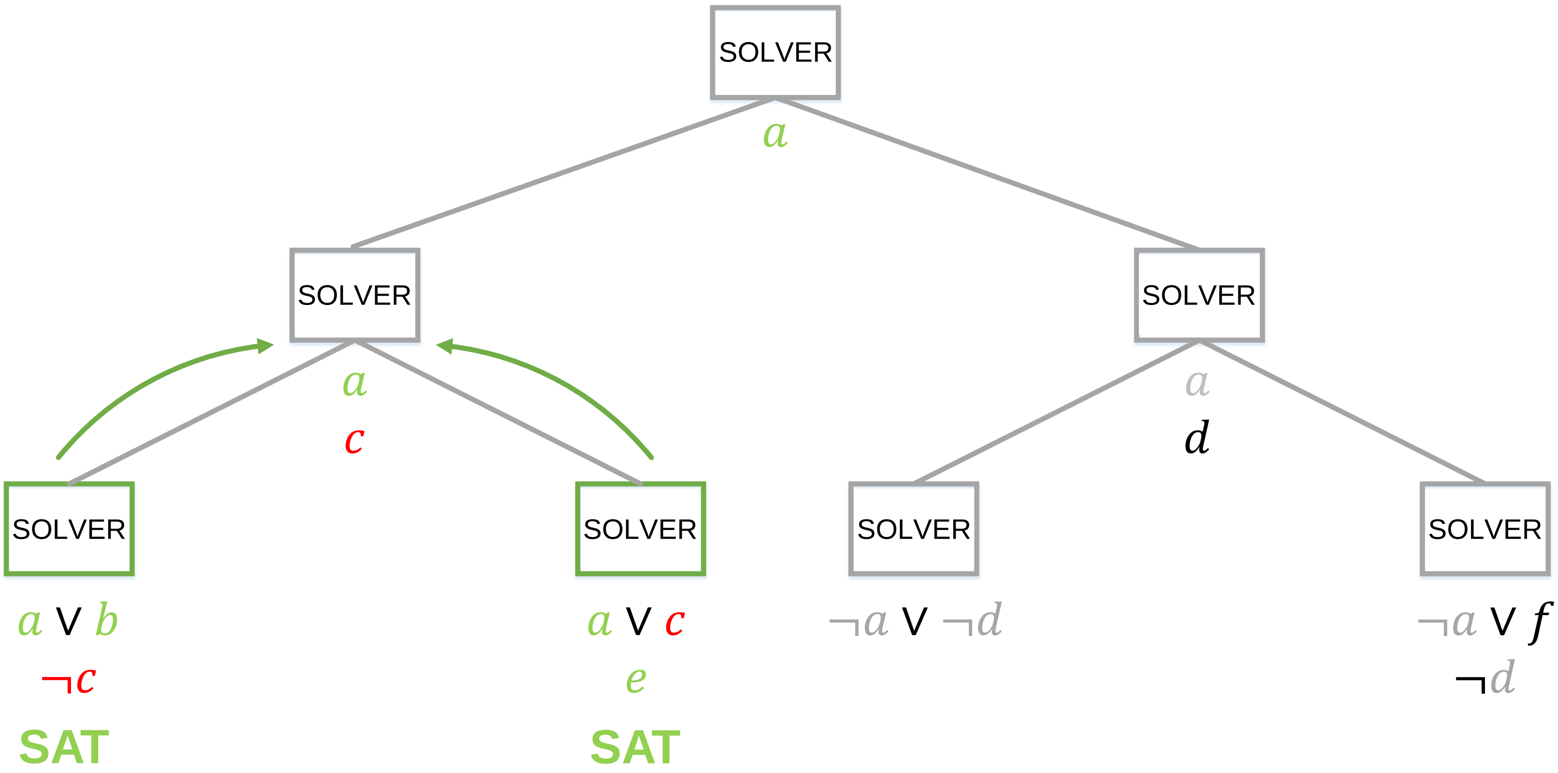
# Example


SOLVER
a
SOLVER
a
c
SOLVER
a
d
SOLVER
a ∨ b
¬c
Decision
SOLVER
a ∨ c
e
Decision
SOLVER
¬a ∨ ¬d
SOLVER
¬a ∨ f
¬d

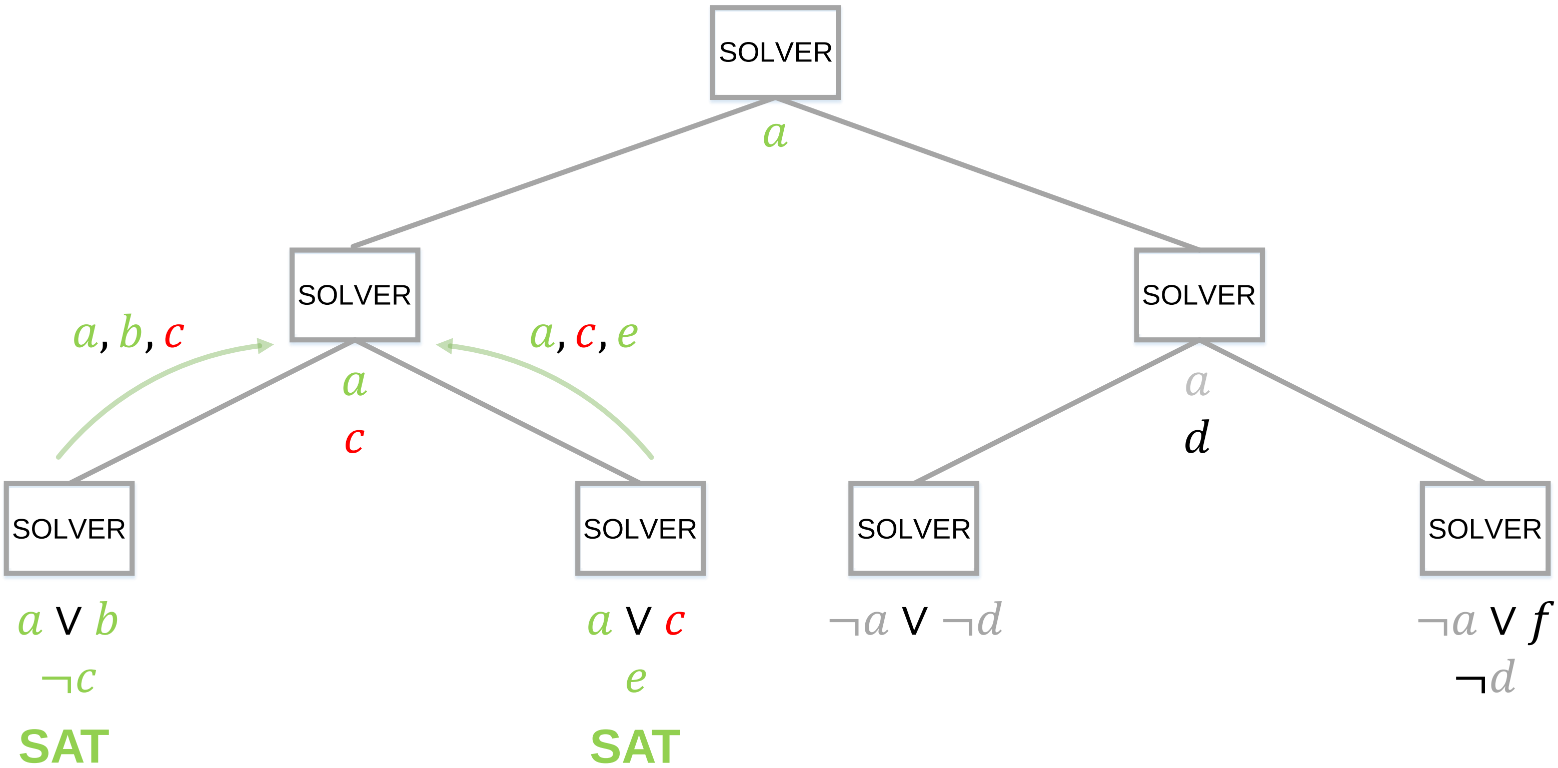
# Example


SOLVER
$a$
SOLVER
$a$
$c$
SOLVER
$a$
$d$
SOLVER
$a \lor b$
$\neg c$
SOLVER
$a \lor c$
$e$
SOLVER
$\neg a \lor \neg d$
SOLVER
$\neg a \lor f$
$\neg d$

# Example


SOLVER
$a$
SOLVER
$a$
$c$
SOLVER
$a$
$d$
SOLVER
$a \lor b$
$\neg c$
SAT
SOLVER
$a \lor c$
$e$
SAT
SOLVER
$\neg a \lor \neg d$
SOLVER
$\neg a \lor f$
$\neg d$

# Example

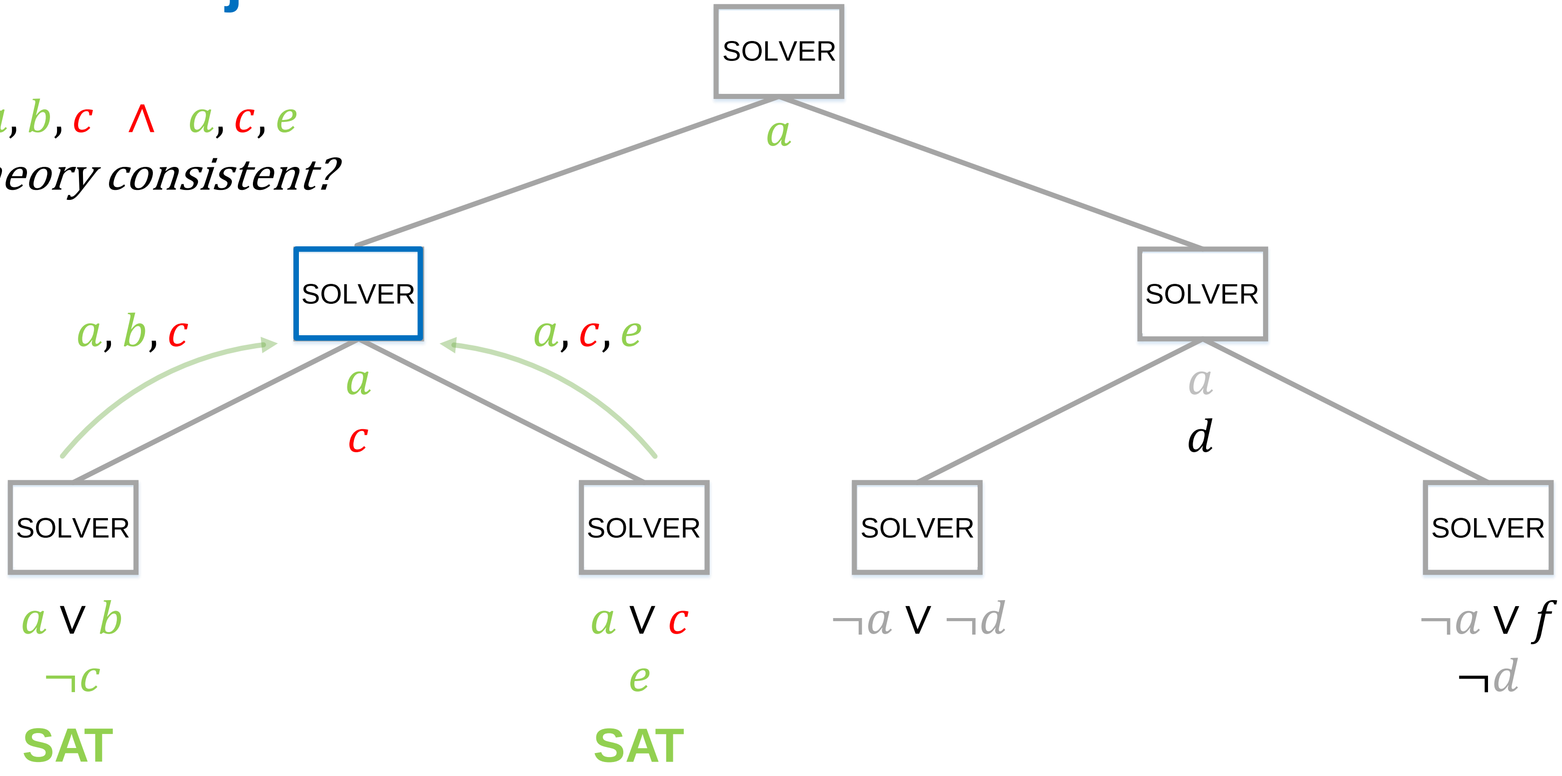

SOLVER
a
SOLVER
SOLVER
a, b, c
a, c, e
a
c
a
d
SOLVER
SOLVER
SOLVER
SOLVER
a ∨ b
¬c
SAT
a ∨ c
e
SAT
¬a ∨ ¬d
¬a ∨ f
¬d

# Example

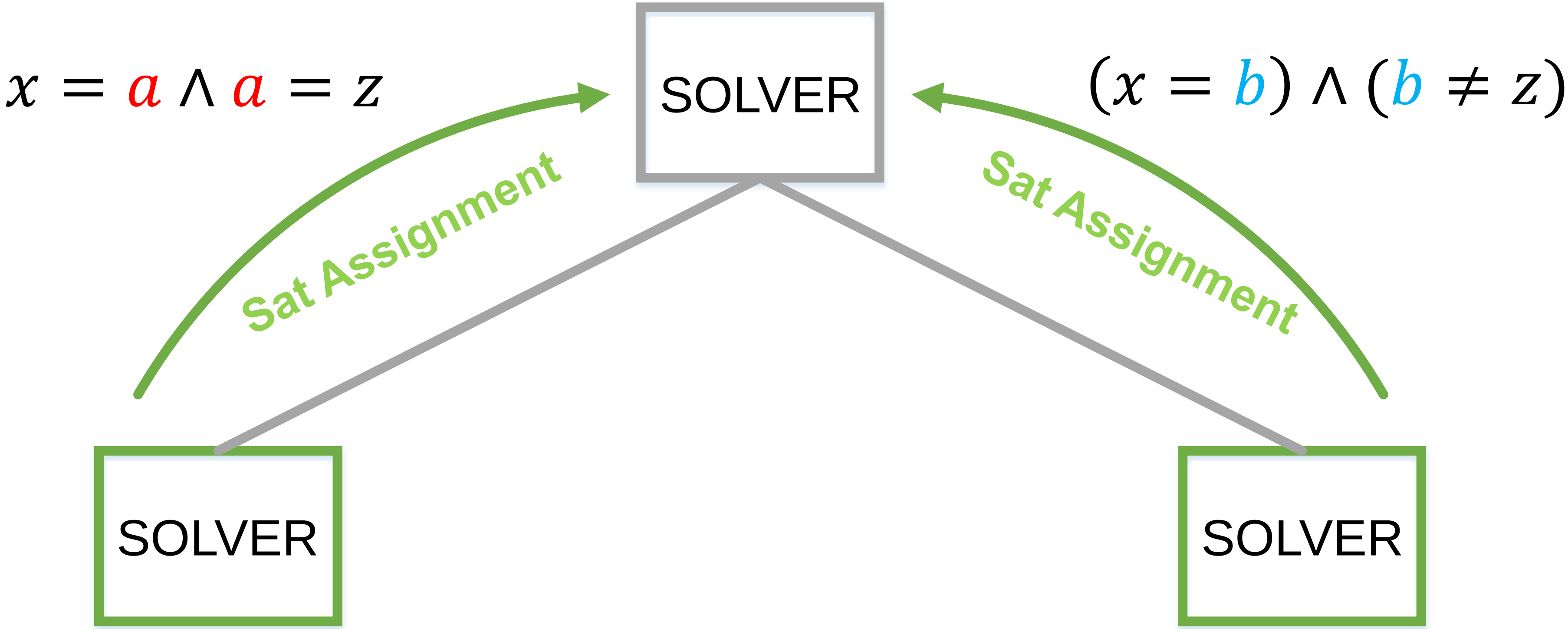
## Theory check of conjunction

$a, b, c \quad \wedge \quad a, c, e$

*Theory consistent?*

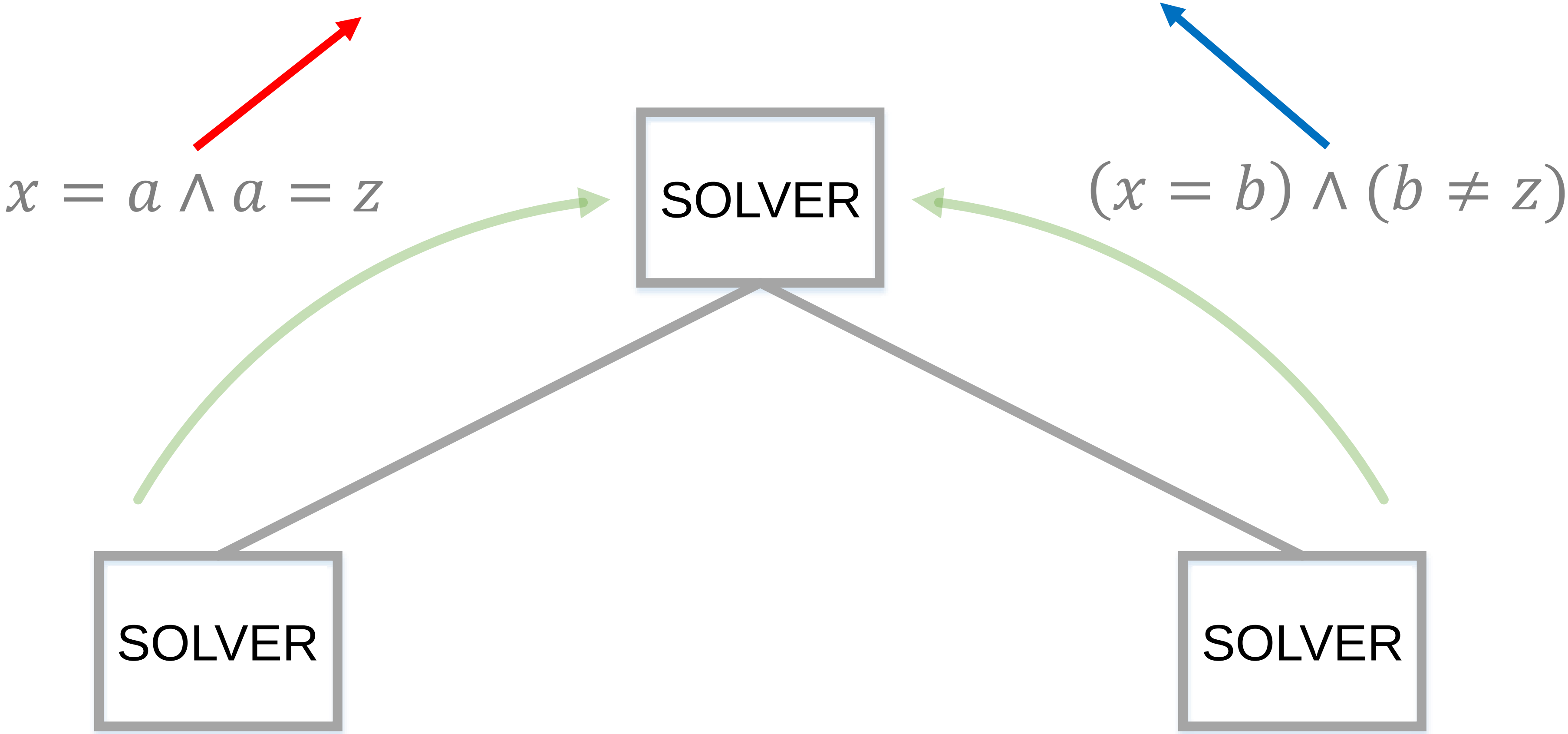


# Theory Check



# Theory Check

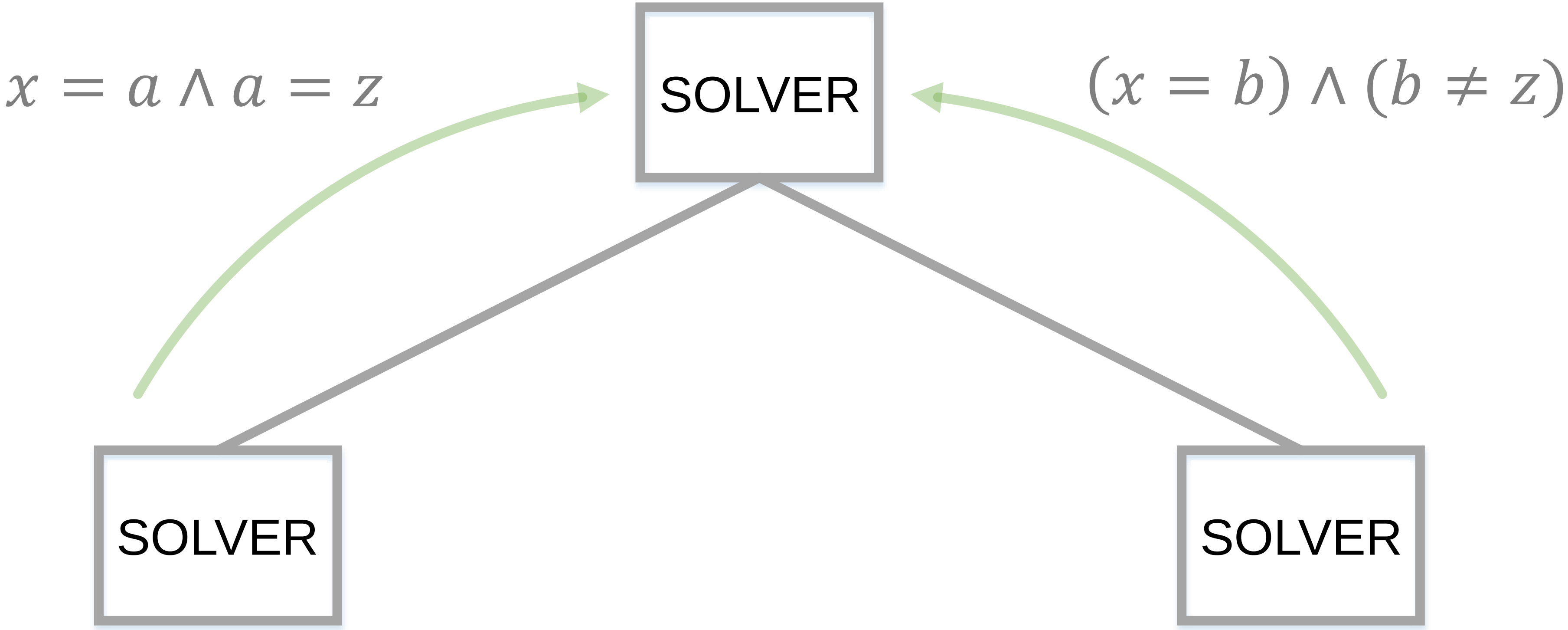
$$(x = a) \land (a = z) \quad \land \quad (x = b) \land (b \neq z)$$



# Theory Check

$$(x = a) \land (a = z) \quad \land \quad (x = b) \land (b \neq z)$$

## Theory Check



# Theory Check

$$(x = a) \wedge (a = z) \quad \wedge \quad (x = b) \wedge (b \neq z)$$

**Theory Inconsistent!**

SOLVER

**Because**

$(x = z) \wedge (x \neq z)$

**Possible Solution:**

Blocking clause

$(a \neq b)$

**BUT:**

Blocking clause

$(a \neq b)$

is not colorable!

SOLVER

SOLVER

# Craig Interpolation

$$CNF(\Phi) = \underbrace{C_1 \wedge C_2 \wedge C_3 \wedge \cdots}_{A} \wedge \underbrace{C_{n-1} \wedge C_n}_{B} = \bot$$

**Interpolant *I*:**

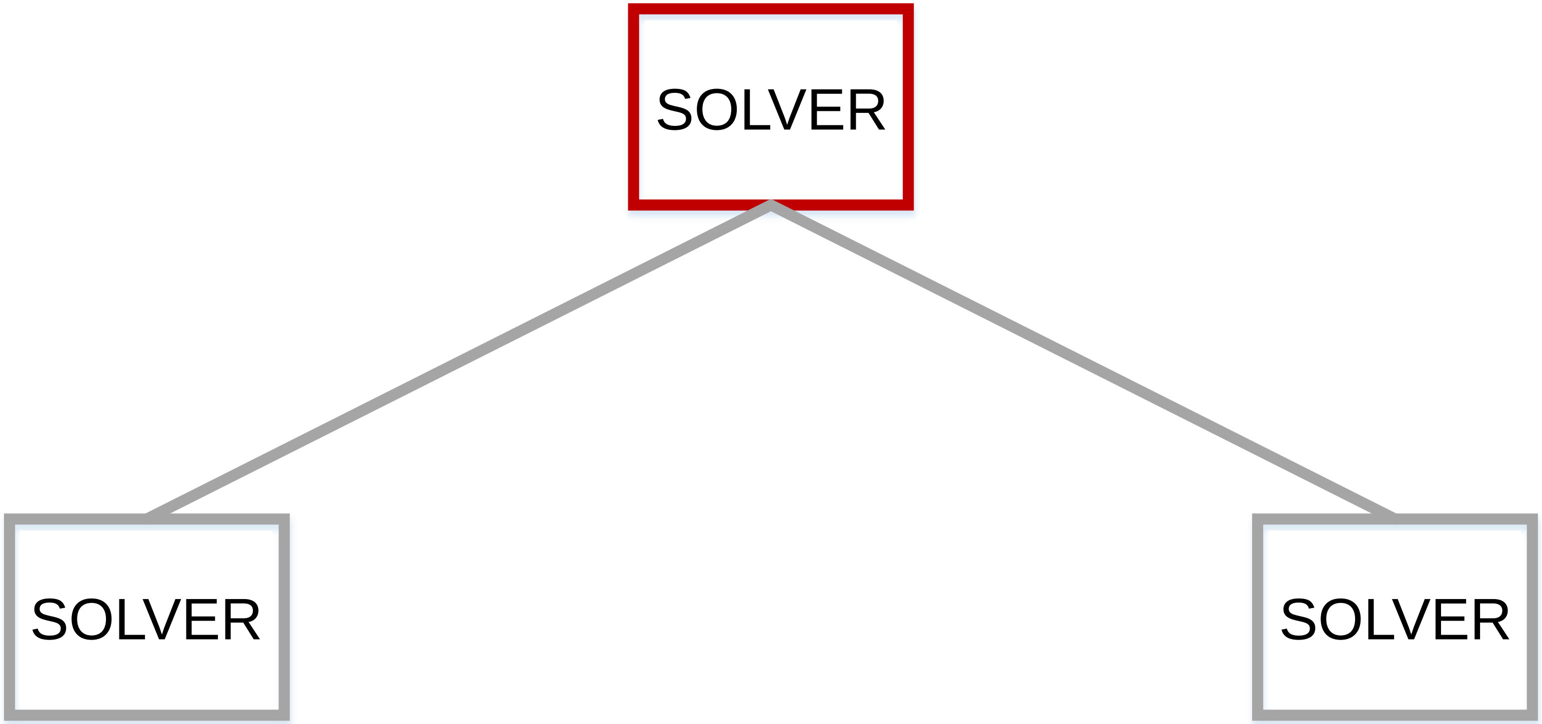
- $A \rightarrow I$
- $I \rightarrow \neg B$, in other words: $I \wedge B = \bot$
- $V(I) \subseteq V(A) \cap V(B)$

**Interpolant contains only global symbols.**

# Interpolation

$$(x = a) \land (a = z) \quad \land \quad (x = b) \land (b \neq z)$$

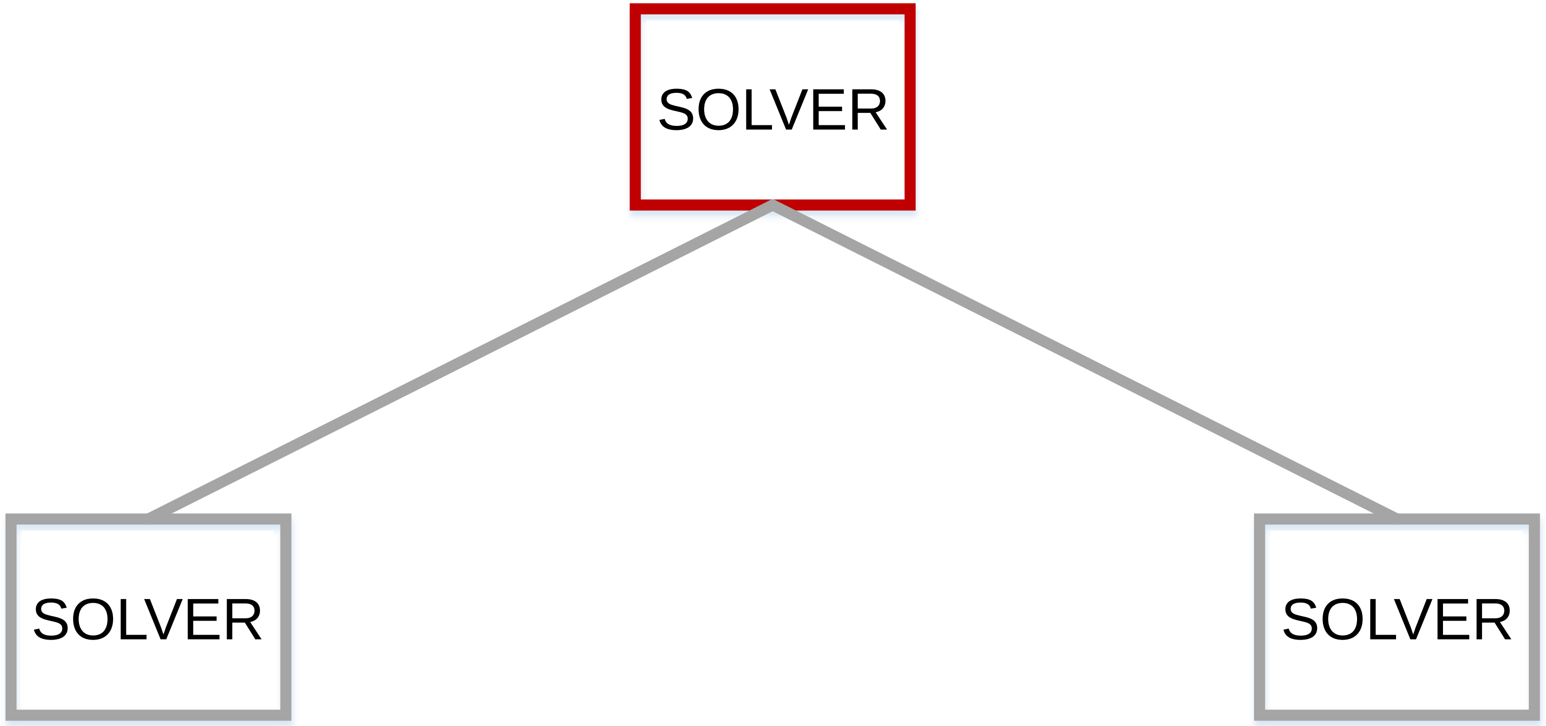
**Interpolate**



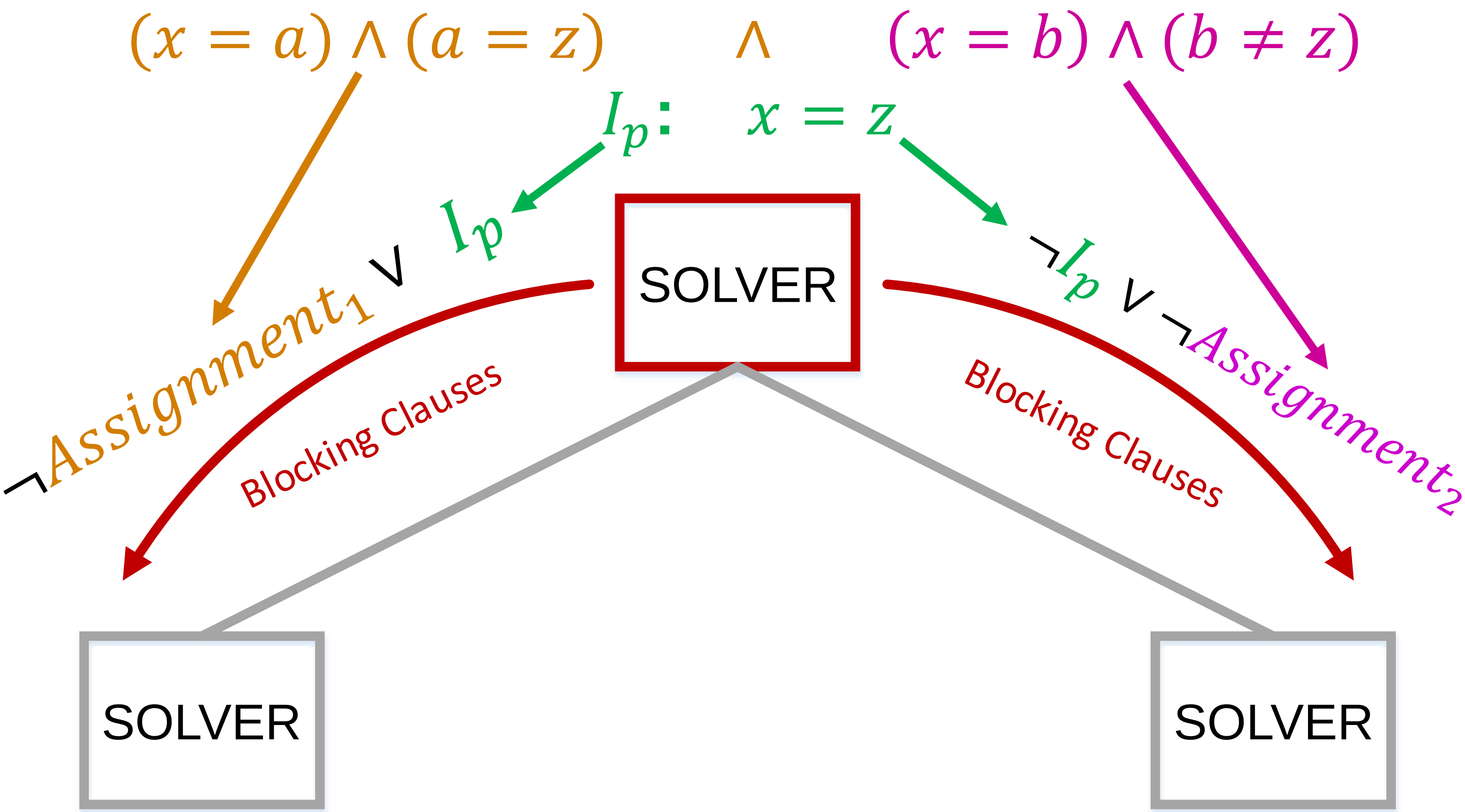
# Interpolation

$$(x = a) \land (a = z) \quad \land \quad (x = b) \land (b \neq z)$$

**Interpolant $I_p$:** $x = z$



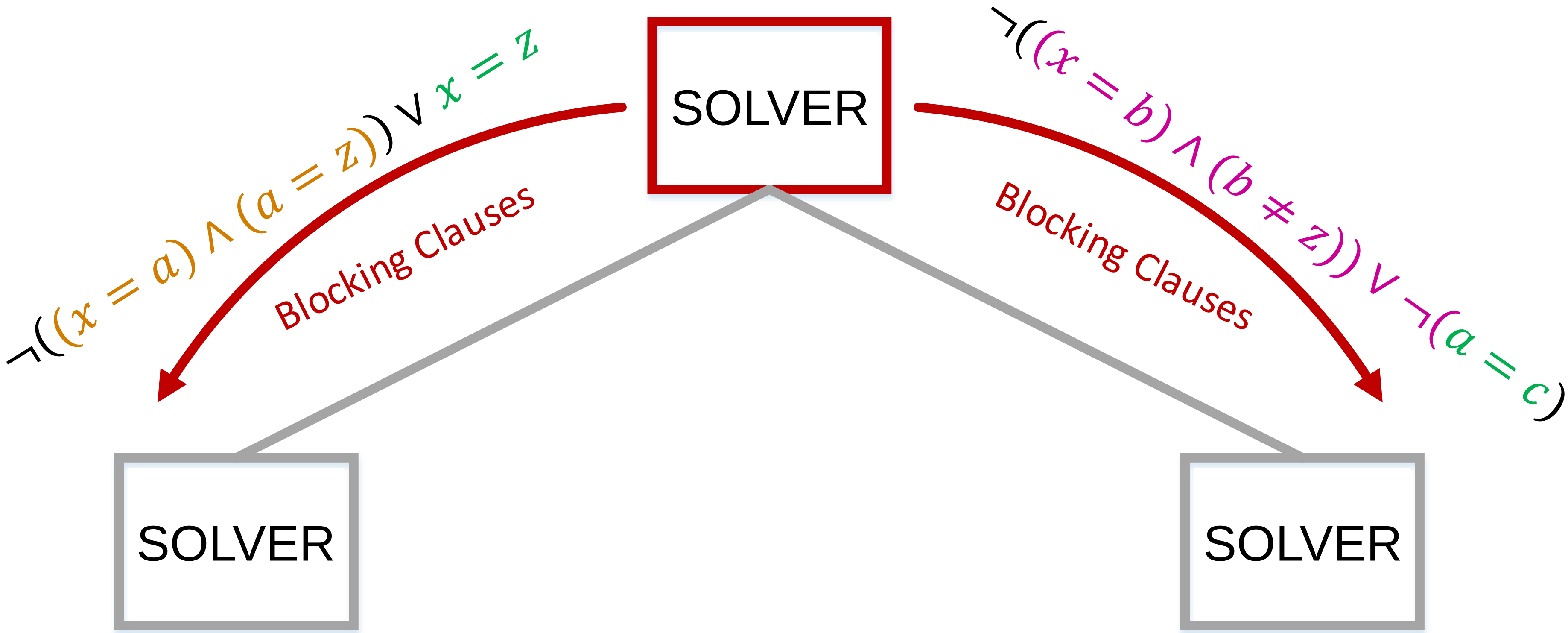
# Interpolation



# Interpolation

$$(x = a) \land (a = z) \quad \land \quad (x = b) \land (b \neq z)$$

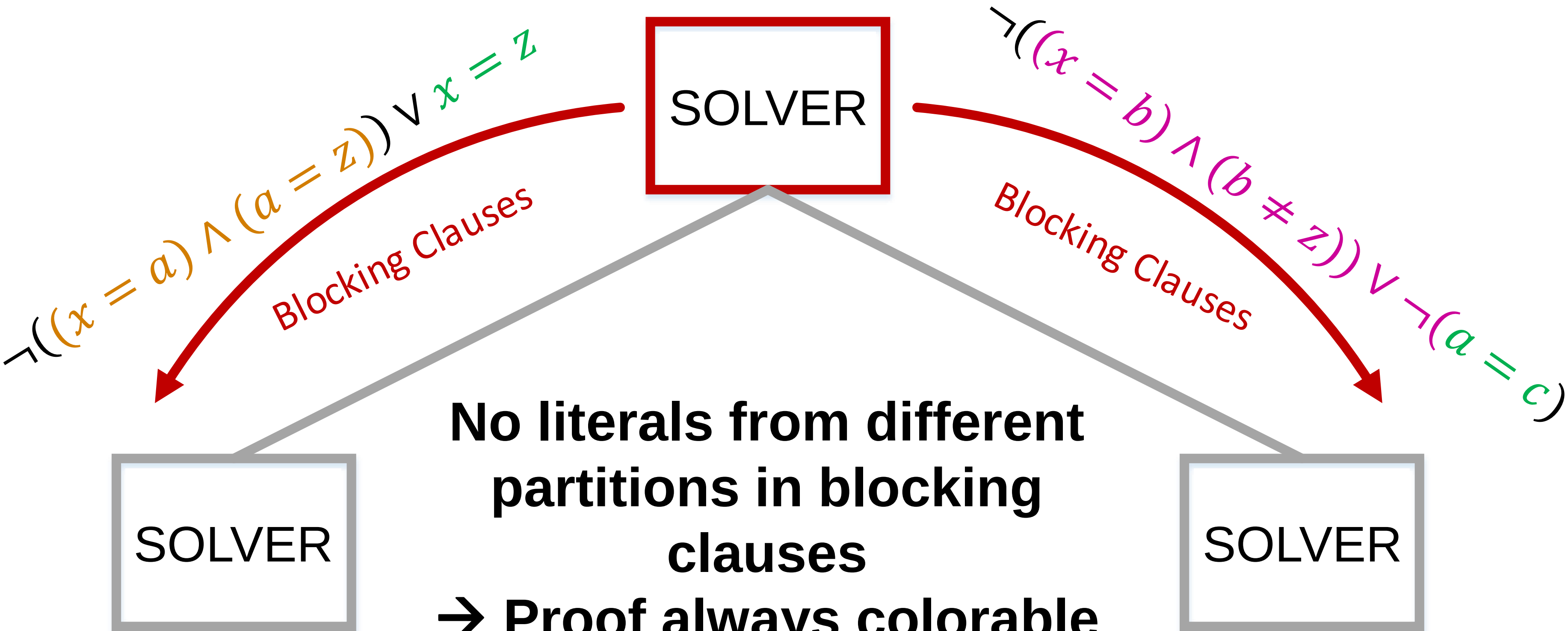
$$I_p: \quad x = z$$



# Interpolation

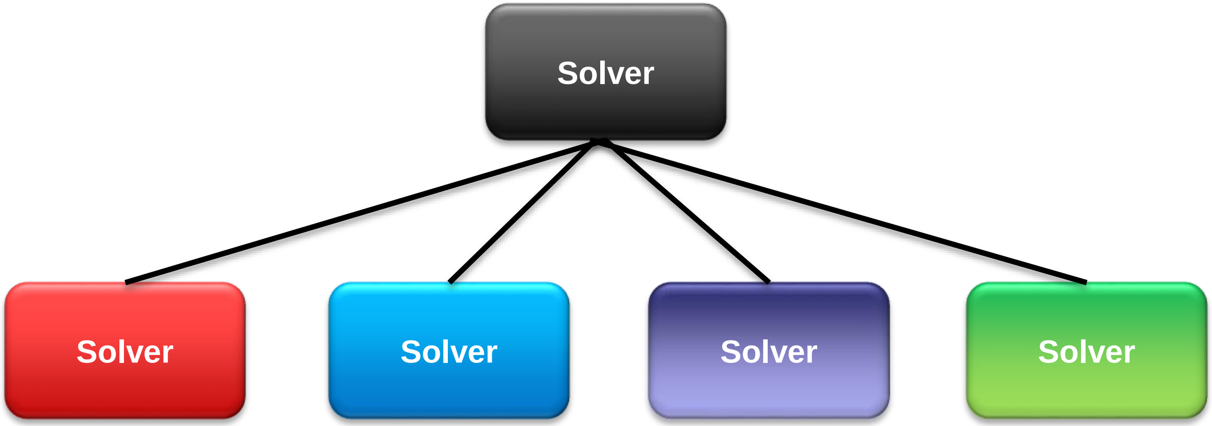
$$(x = a) \wedge (a = z) \quad \wedge \quad (x = b) \wedge (b \neq z)$$

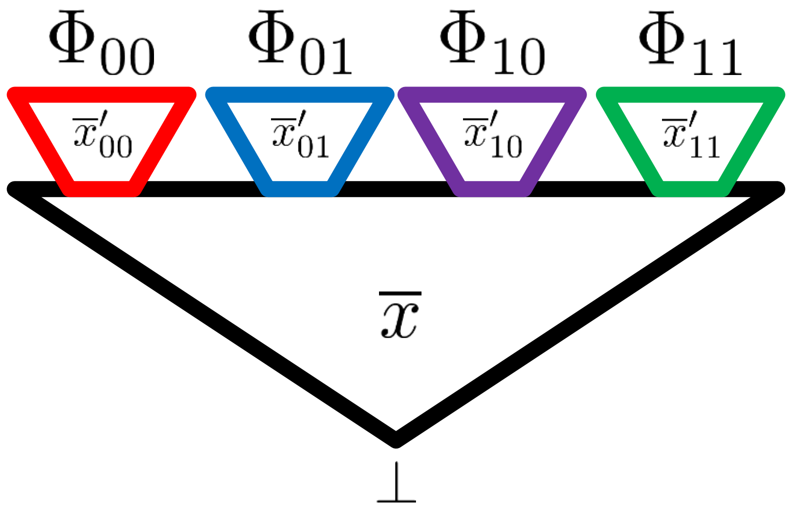
$$I_p\text{:} \quad x = z$$




**No literals from different partitions in blocking clauses**
**→ Proof always colorable**

# Proof production




Root node resolves only over global literals

Premises of proof in root node are proofs of child nodes

# Current State & Outlook

- Prototype implemented („Proof of concept“) with MiniSat + MathSAT
- Relativly good runtime but much optimisation potential...
- Currently implementing proof production.

# Conclusion

- Modular SMT Solving
  - Colorable and local-first proof directly from SMT solver.
  - Possible for all theories with interpolants in same theory.
- Craig Interpolation
  - Produces colorable blocking clauses
  - Multiple coordinated interpolants from just one proof
- Therefore the world is now a slightly better place ☺

# Thank You!

Questions?

# Appendix

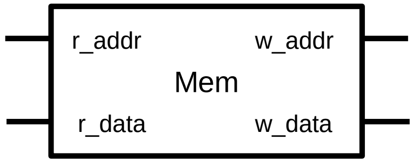
# Specification

# Specification

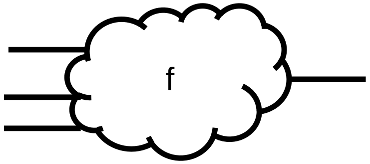
**Correctness:** First-Order Logic Formula $\Phi$

Important Building Blocks:

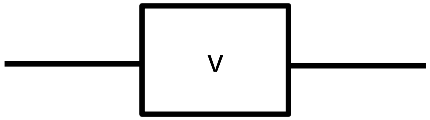
- Array Variables
  - Addressable Memories




- Uninterpreted Functions & Predicates
  - Combinational Circuits




- Domain Variables
  - Single Element Storage
  - Primary Inputs/Outputs



# Certificate via Interpolation

# Certificate via Interpolation

$$\Psi = \forall\, mem,\ \ reg,\ \ pipelinestate\ .$$
$$\exists\, stall, forward\ .$$
$$\forall\, mem',\ \ reg',\ \ pipelinestate'\ .\quad \Phi$$

- *stall, forward*: **Boolean** control signals
- *mem, reg, pipelinestate*: Uninterpreted domain

Compute **Certificates**:

$$(stall, forward) = f(mem, reg, pipelinestate)$$

# Certificate via Interpolation

- $\Psi \;=\; \forall \vec{a}.\;\; \exists \vec{c}.\; \forall \vec{b}.\; \Phi(\vec{a}, \vec{b}, \vec{c})$
  - **$\Psi$ is valid**

- Function $\vec{c} = \sigma(\vec{a})$

- Such that: $\Phi\left(\vec{a}, \vec{b}, \sigma(\vec{a})\right)$ is **valid**

# Certificate via Interpolation

$$\underbrace{\neg\Phi\left(\vec{a},0,\vec{b}_0\right)}_{A} \wedge \underbrace{\neg\Phi\left(\vec{a},1,\vec{b}_1\right)}_{B} = \bot$$

*0 not allowed* (A) — *1 not allowed* (B)

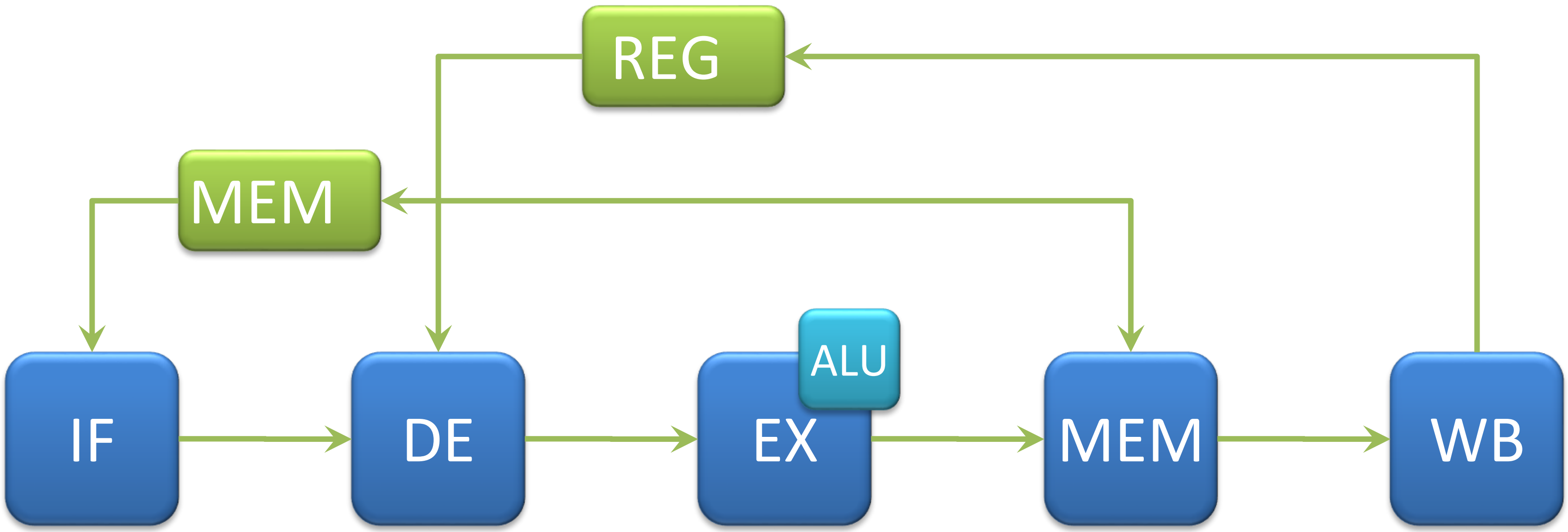
**Interpolant $I(\vec{a})$:**

- $\neg\Phi\left(\vec{a},0,\vec{b}_0\right) \rightarrow I$
  - $I$ is 1, whenever 0 not allowed
- $I \rightarrow \Phi\left(\vec{a},1,\vec{b}_1\right)$
  - Whenever $I$ is 1, 1 is allowed

**Boolean Case: see Jiang et al., ICCAD'09**

# Sample Application

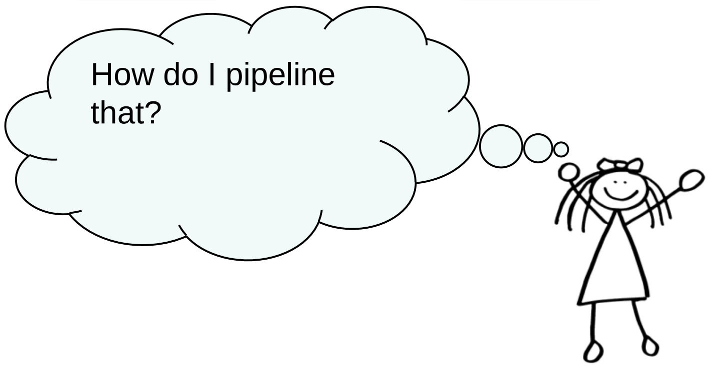
# A Processor



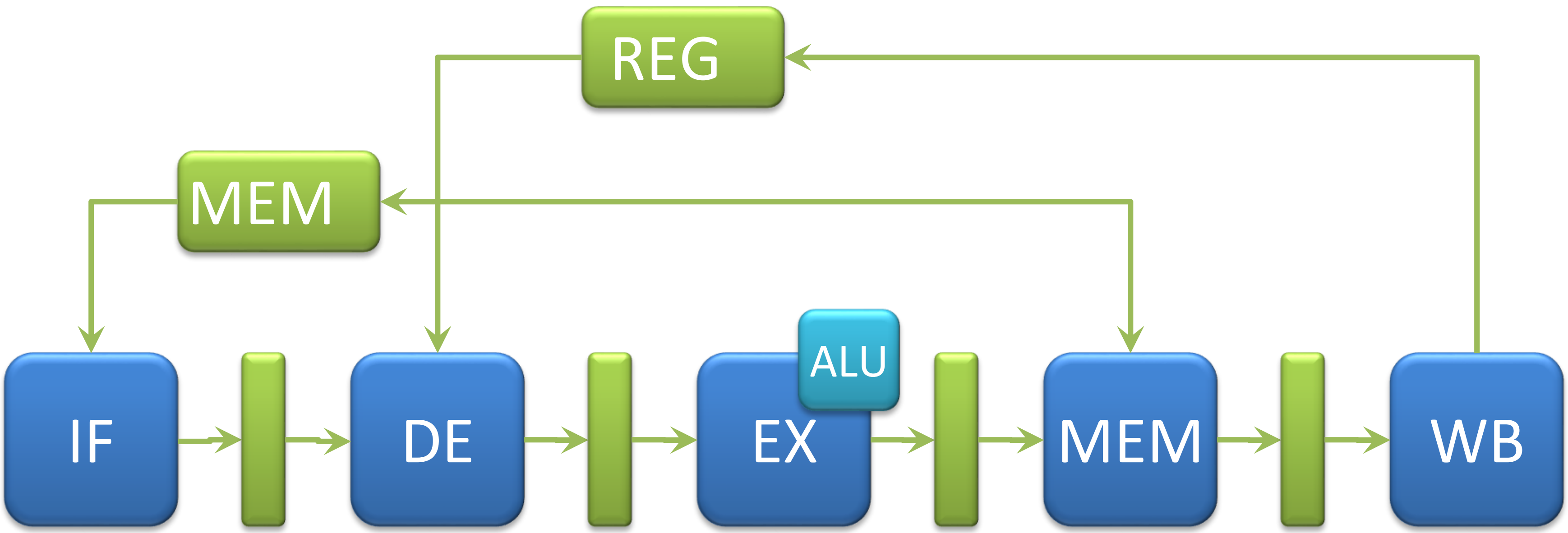

Tough:

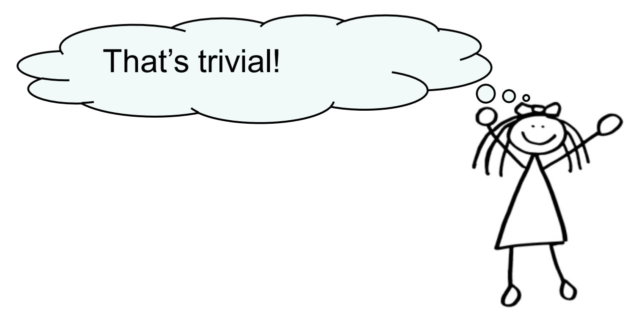
64-bit datapath

very complex arithmetic logic unit

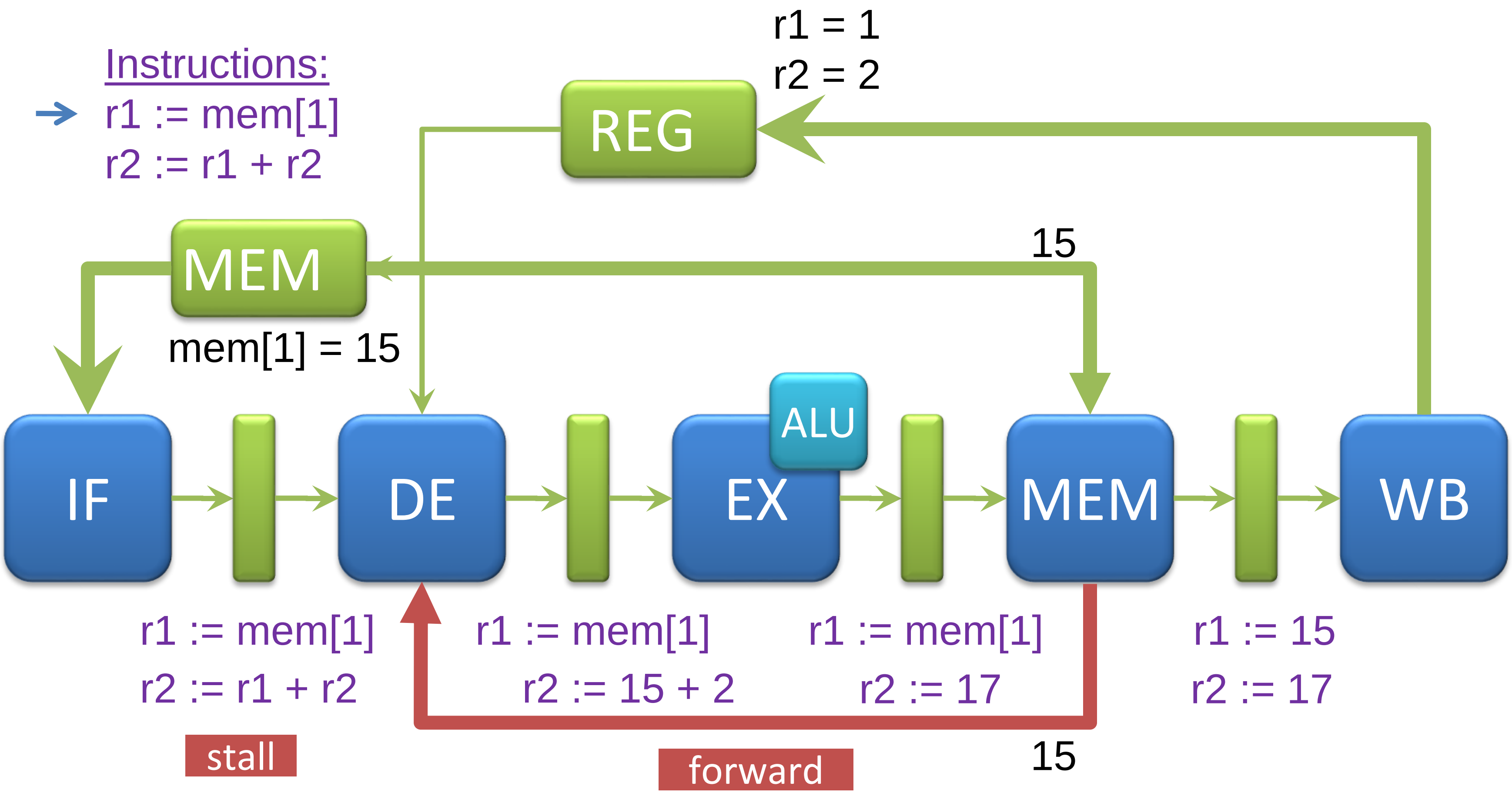


# A Pipelined Processor

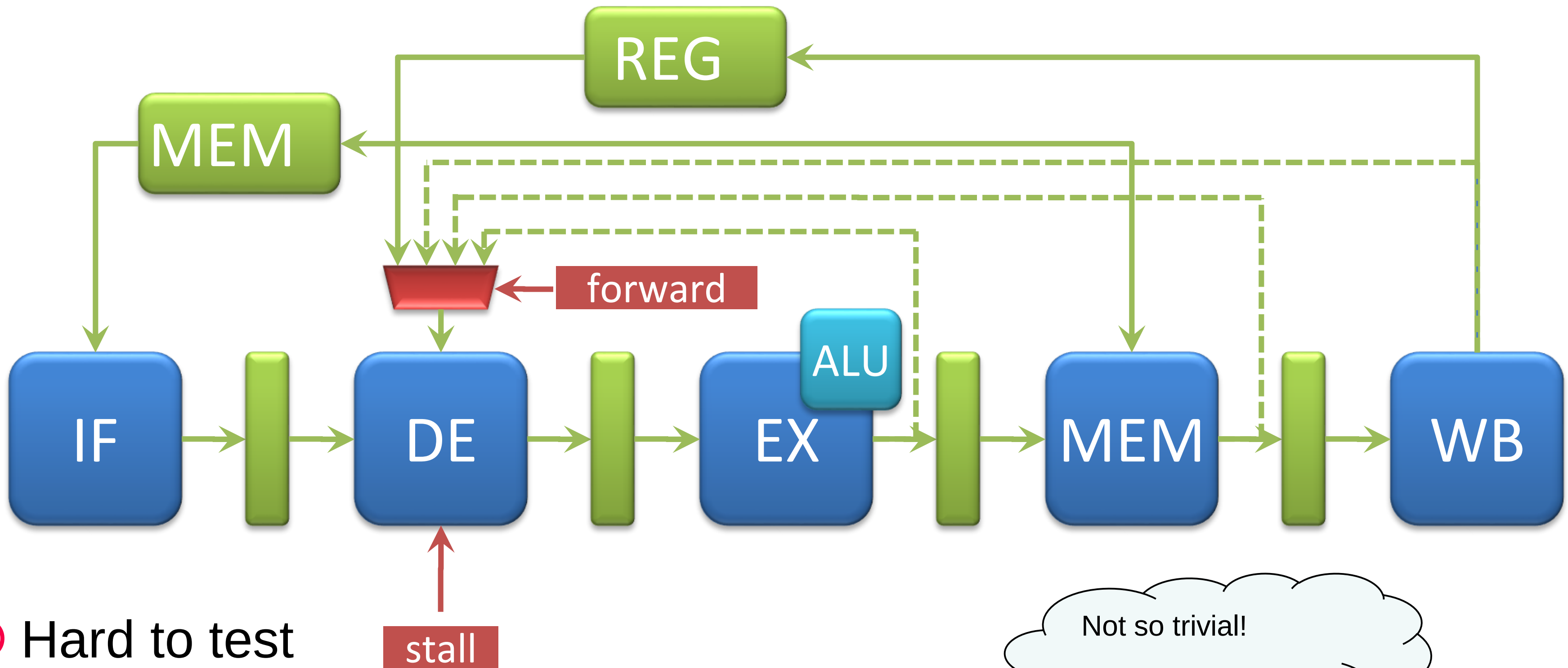


# A Pipelined Processor

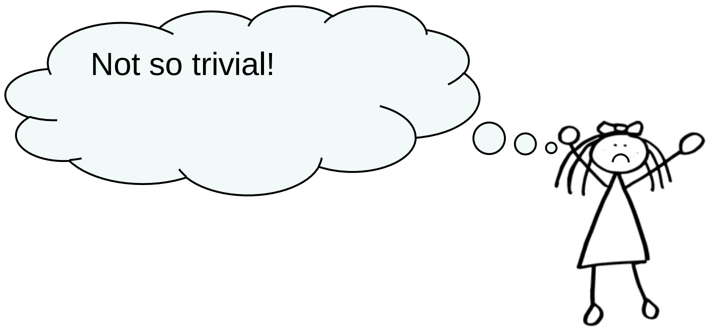


# A Pipelined Processor




- ☹ Hard to test
- ☹ Hard to implement
- ☺ Easy to specify

→ Burch-Dill paradigm



# Craig Interpolation

# Craig Interpolation

$$CNF(\Phi) = \underbrace{C_1 \wedge C_2 \wedge C_3 \wedge \cdots}_{A} \wedge \underbrace{C_{n-1} \wedge C_n}_{B} = \bot$$

## Interpolant *I*:

- $A \rightarrow I$
- $I \rightarrow \neg B$, in other words: $I \wedge B = \bot$
- $V(I) \subseteq V(A) \cap V(B)$