Beyond SAT and BDD based model checking

Author(s):
Biere, Armin

Publication Date:
2001

Permanent Link:
https://doi.org/10.3929/ethz-a-004242413 →

Beyond SAT and BDD based Model Checking

Armin Biere

Institute of Computer Systems
Department of Computer Science
ETH Zürich, Switzerland

http://www.inf.ethz.ch/personal/biere

Workshop on Computer Aided Design and Test - BDDs versus SAT
28.01.2001 – 02.02.2001
Dagstuhl, Germany

Model Checking

BDD based	SAT based
robust algorithms	small class of examples
small designs	larger designs
complete	incomplete

complete – actually verifies properties of interest

incomplete – finds (some) bugs of interest

How to make a SAT based model checker complete?

Checking Safety Properties

1. Assume safety property does not hold
2. Derive partial state assignment
3. Assume initial state has been reached
4. If satisfiable generate counter example trace and exit
5. Remove assumptions from 3 and save assignments for backtracking
6. Save old partial state assignment in Clause Data Base
7. Substitute old assignment in next state equations and goto 2

Model Checking with SAT and ATPG

Partial assignments

Caching with a Clause Data Base

Sequential Justification

Tool Chain

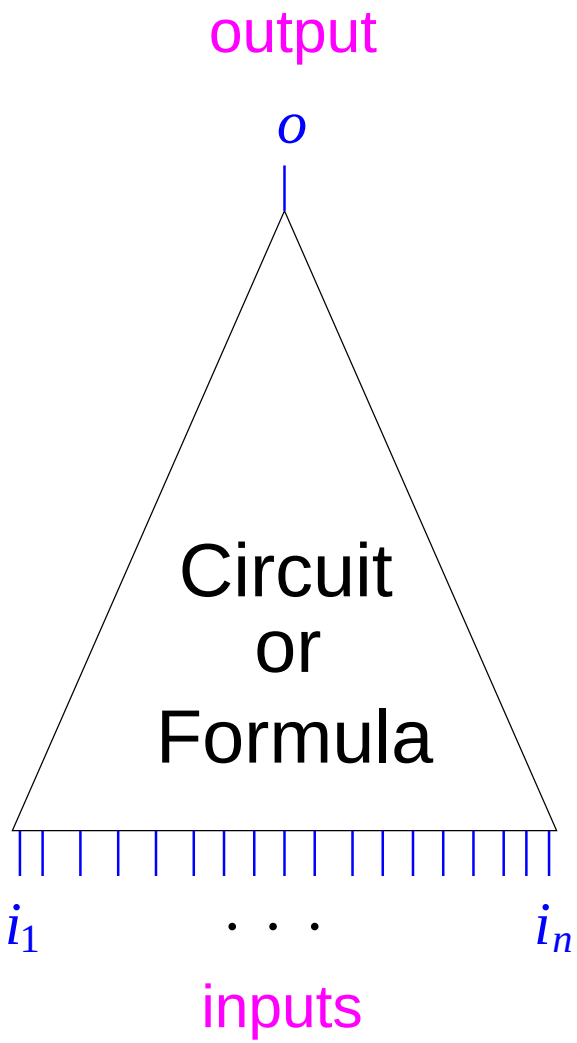
Models

SMVFlatten and FlatSMV

Complete Bounded Model Checker

Related Work

Preliminary Results



SAT

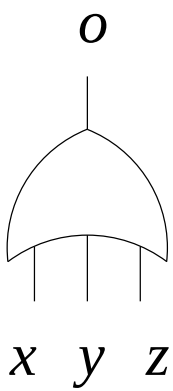
Find assignment to variables $i_1, \ldots i_n$ for which the formula evaluates to true

Justification

Find partial assignment to primary inputs $i_1, \ldots i_n$ that produce a certain logic value at output o

Partial Assignments

Circuit



Formula

$$o \leftrightarrow (x \vee y \vee z)$$

CNF

$$(\neg x \vee o)$$
$$(\neg y \vee o)$$
$$(\neg z \vee o)$$
$$(\neg o \vee x \vee y \vee z)$$

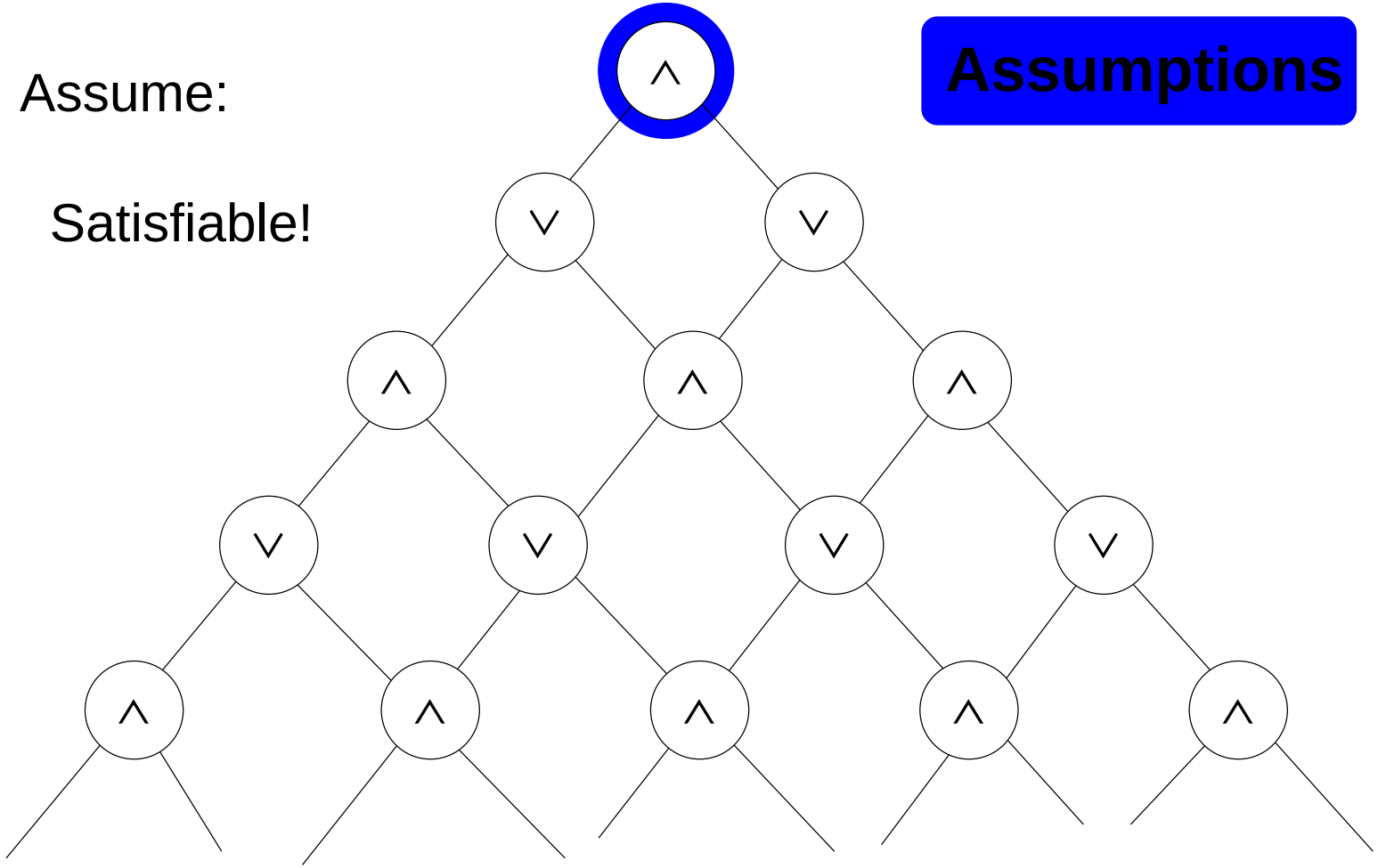
The partial assignment $x = 1$ is sufficient to justify $o = 1$

Values of y and z can be arbitrary!

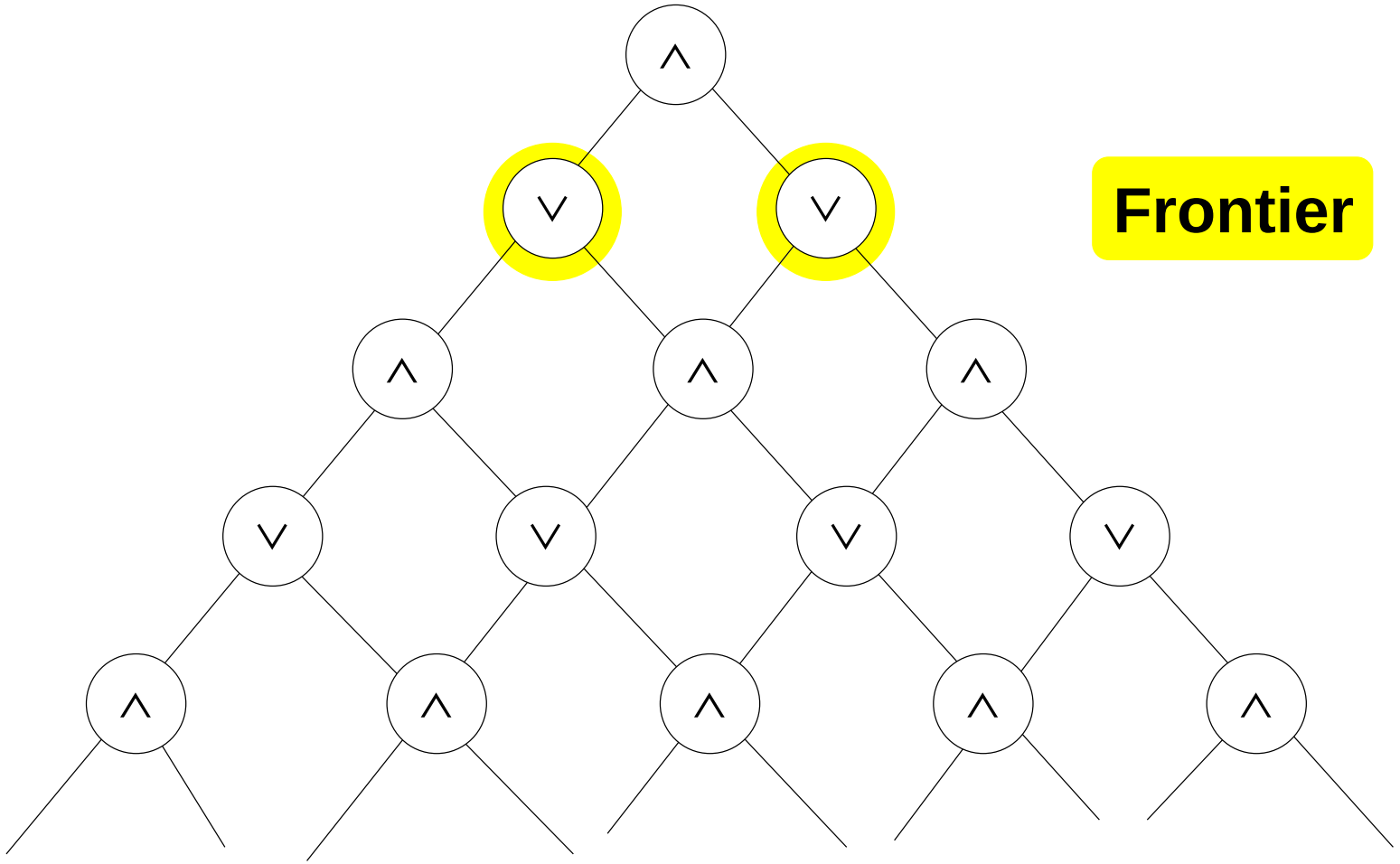
The Concept of Justification Frontiers

Assume:

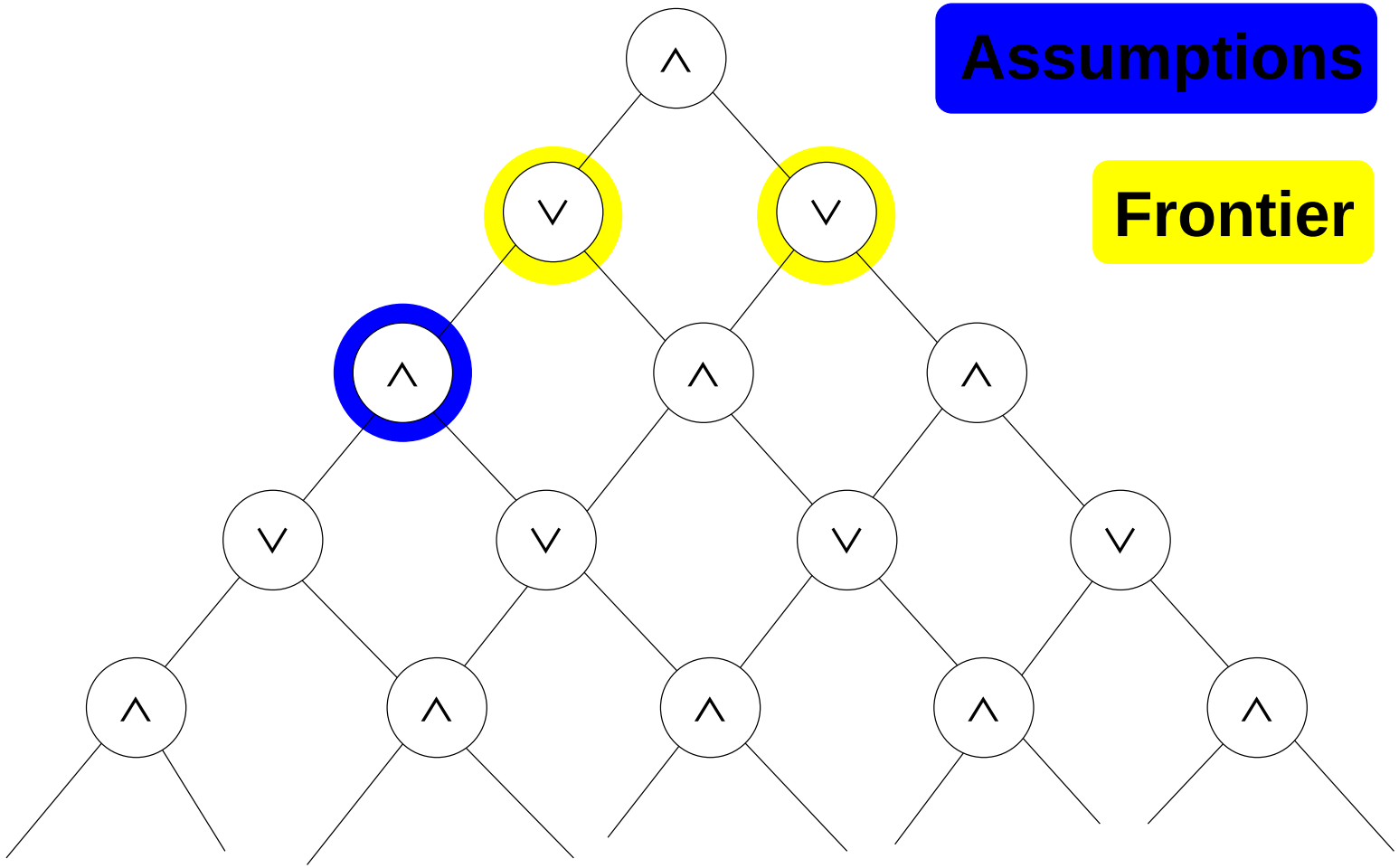
Satisfiable!



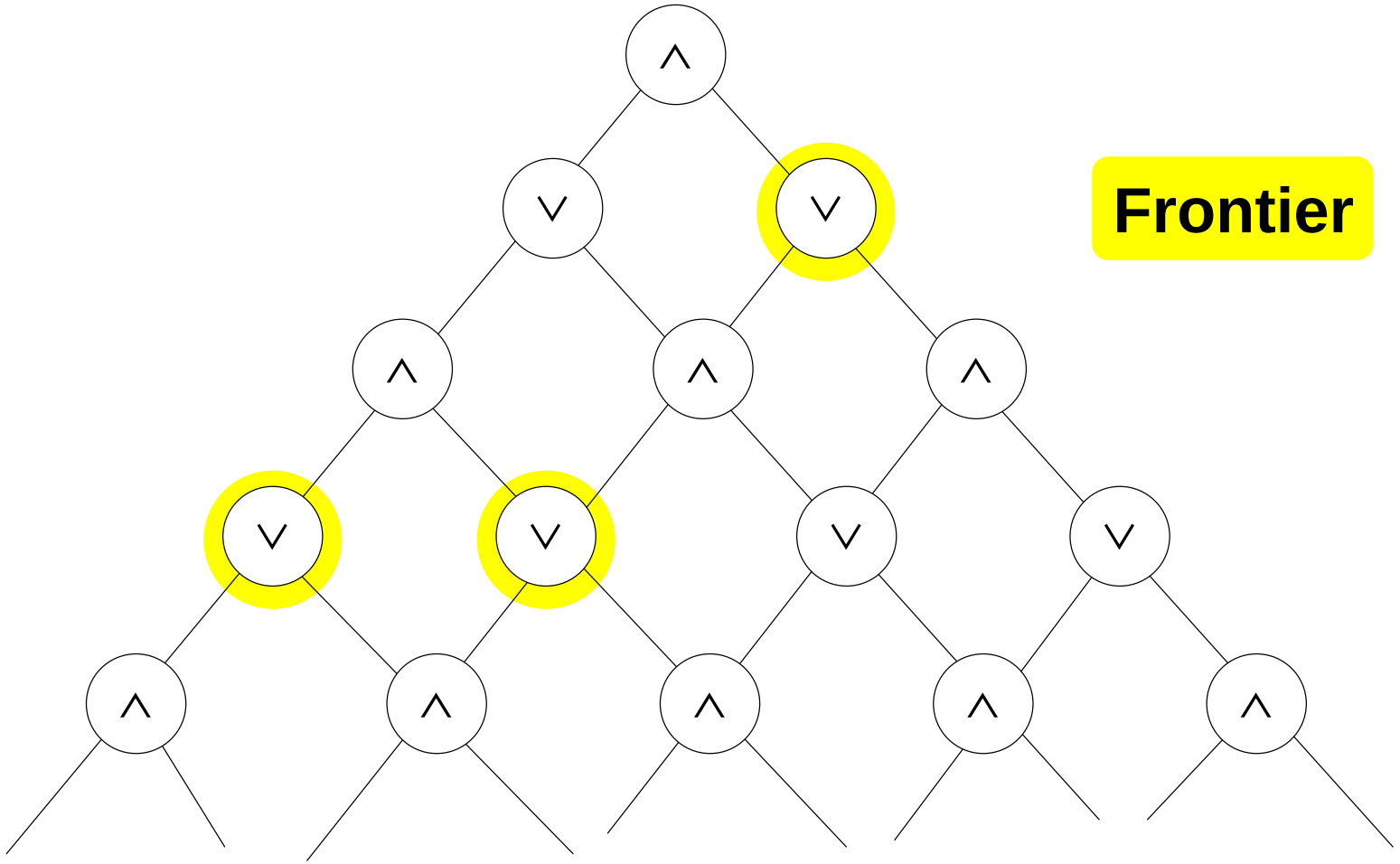
The Concept of Justification Frontiers



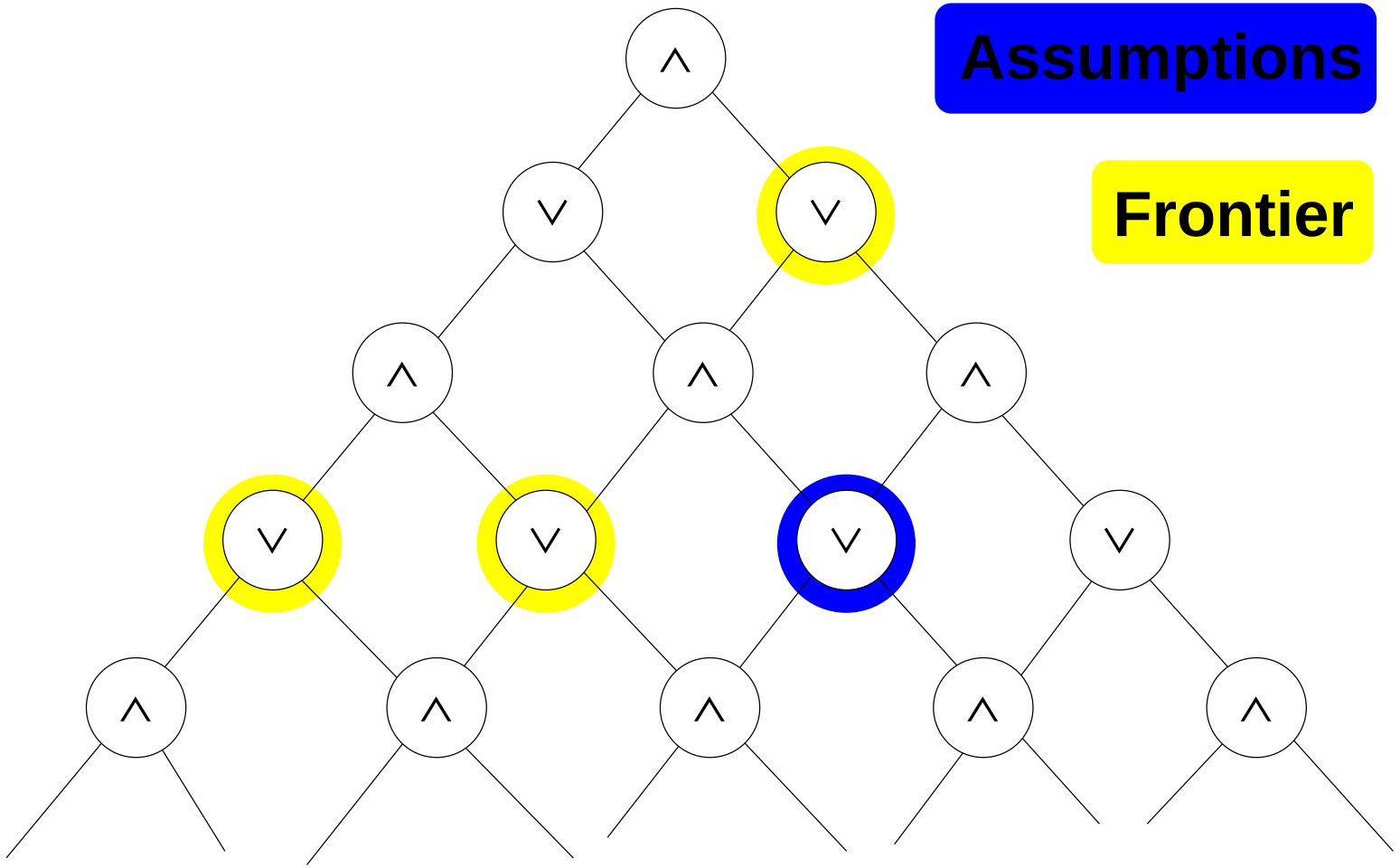
The Concept of Justification Frontiers



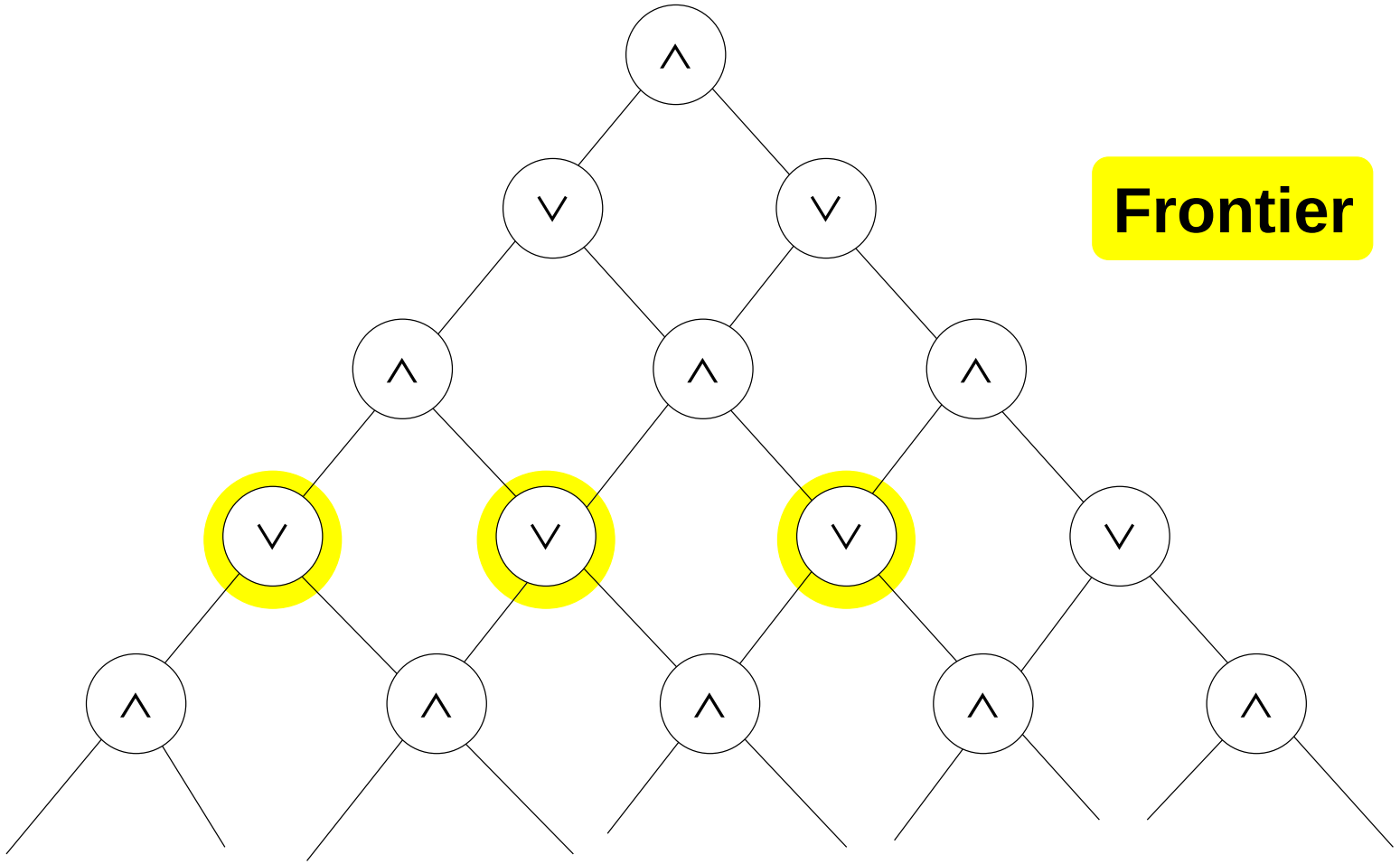
The Concept of Justification Frontiers



The Concept of Justification Frontiers



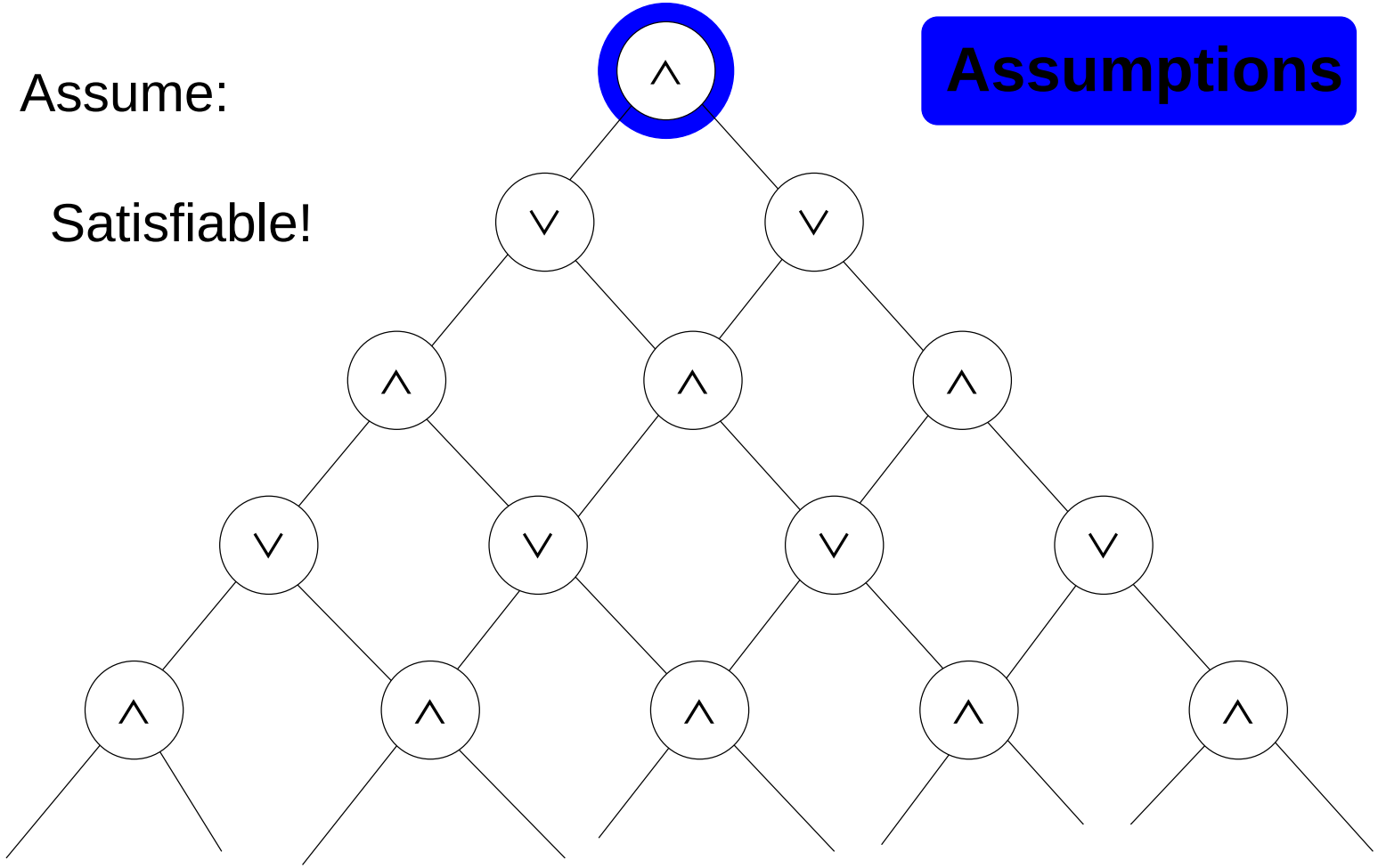
The Concept of Justification Frontiers



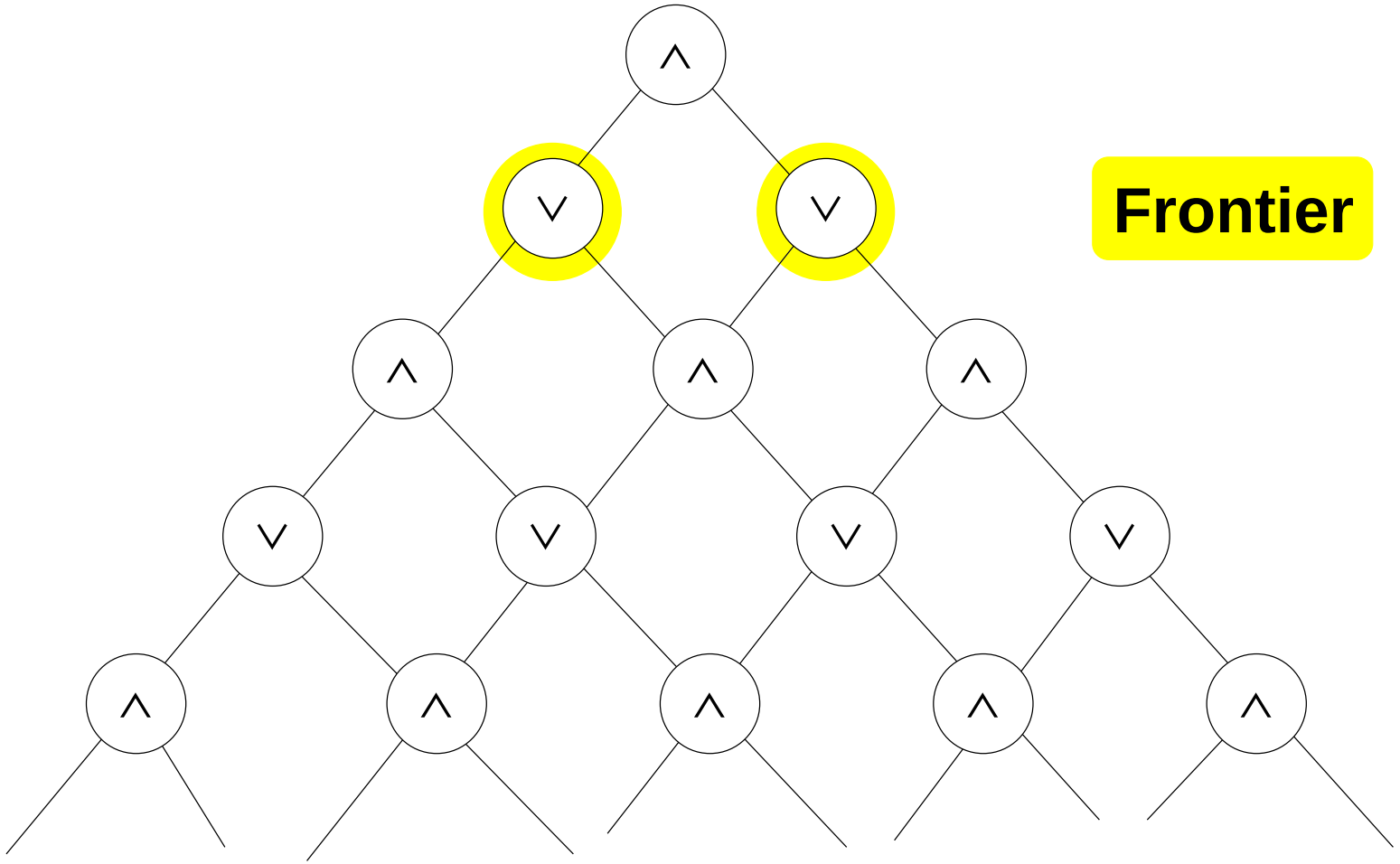
The Concept of Justification Frontiers

Assume:

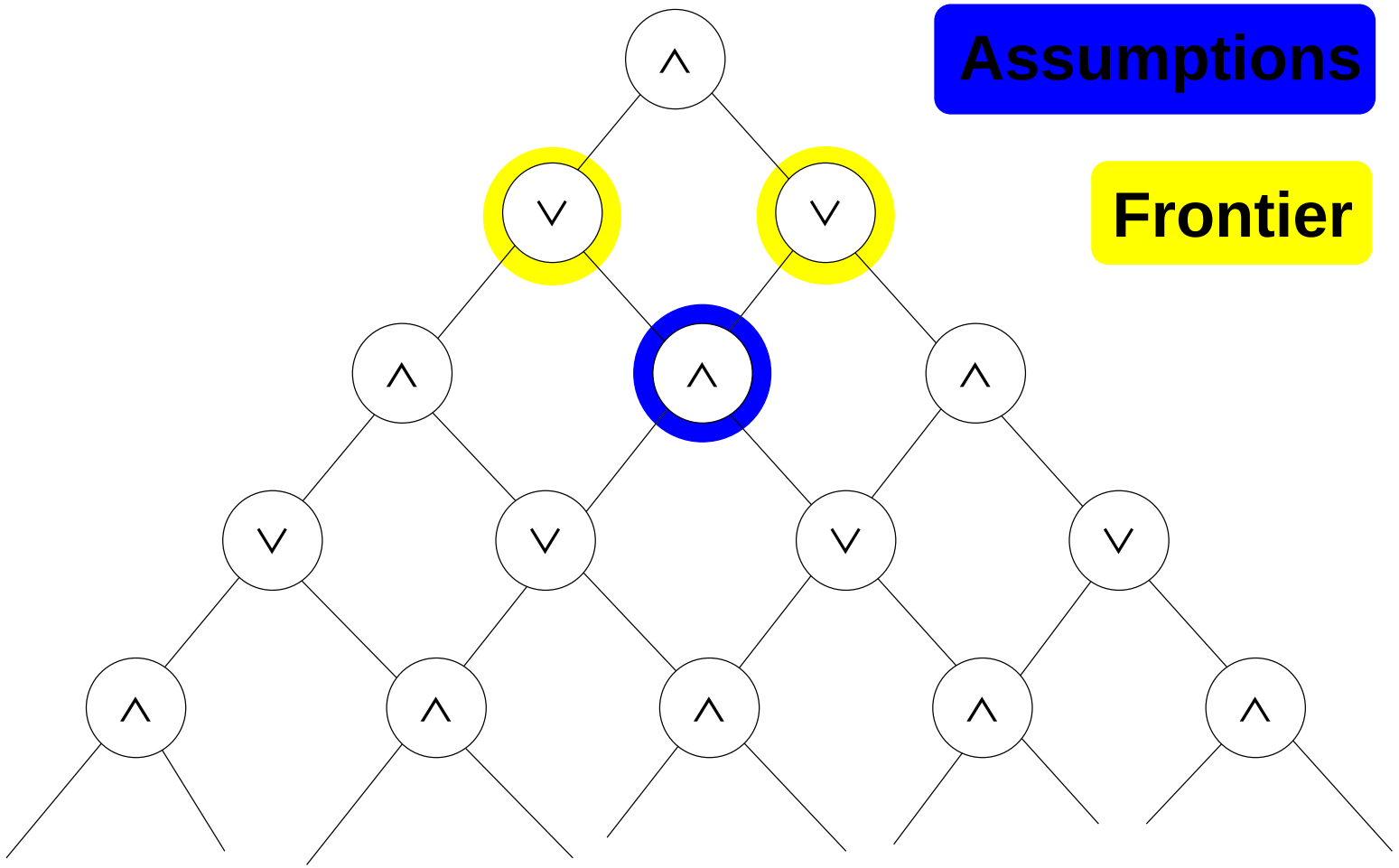
Satisfiable!



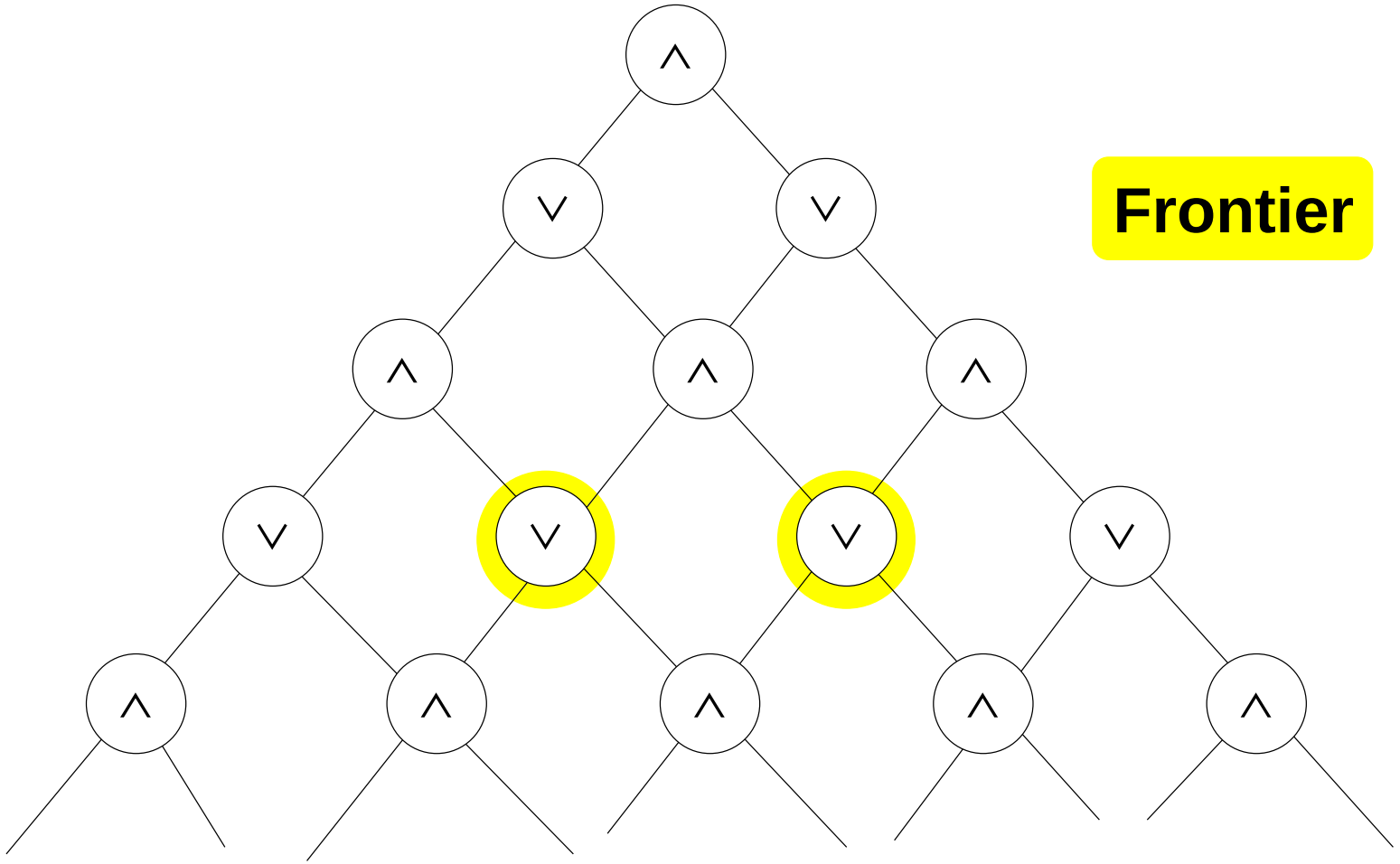
The Concept of Justification Frontiers



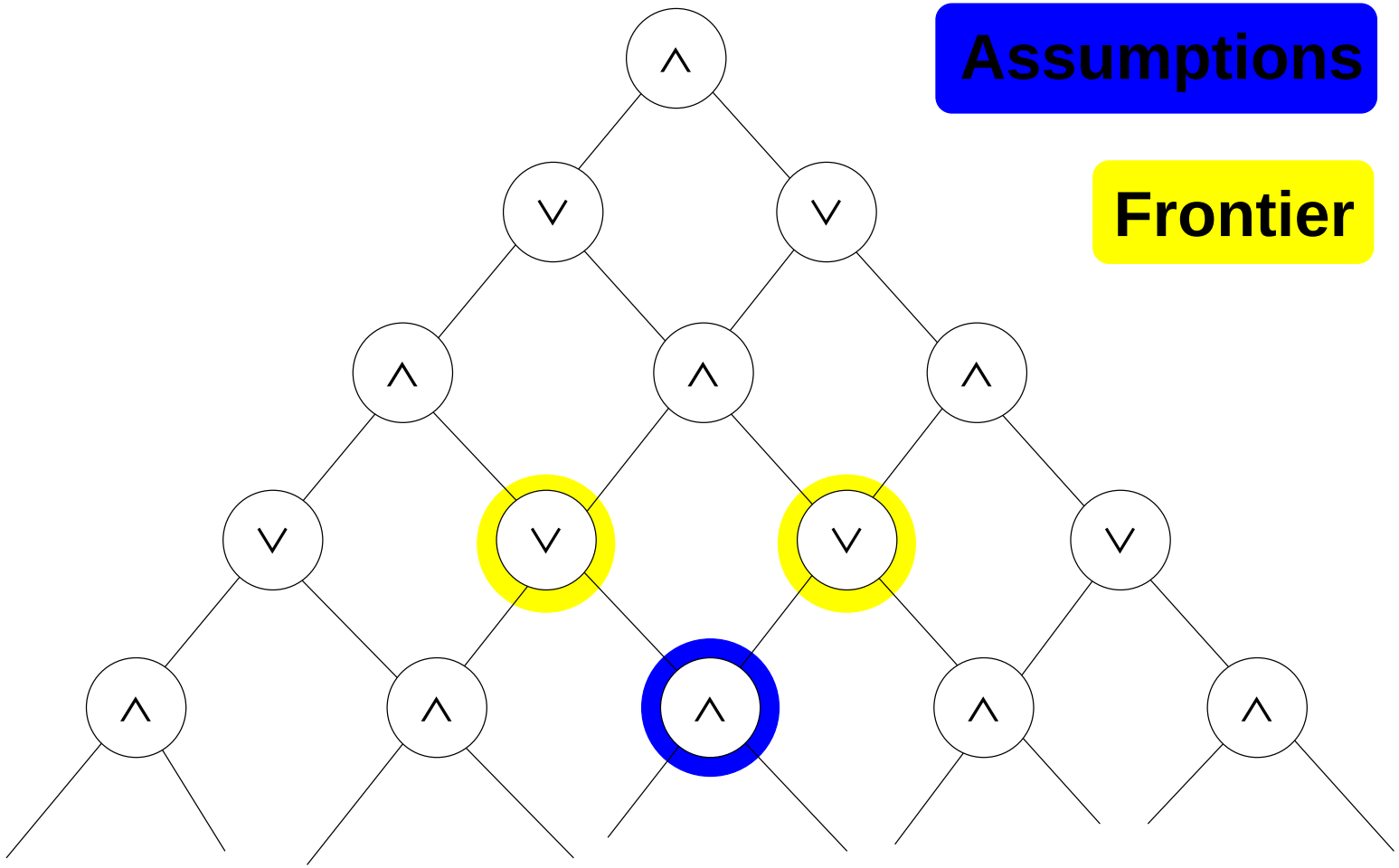
The Concept of Justification Frontiers



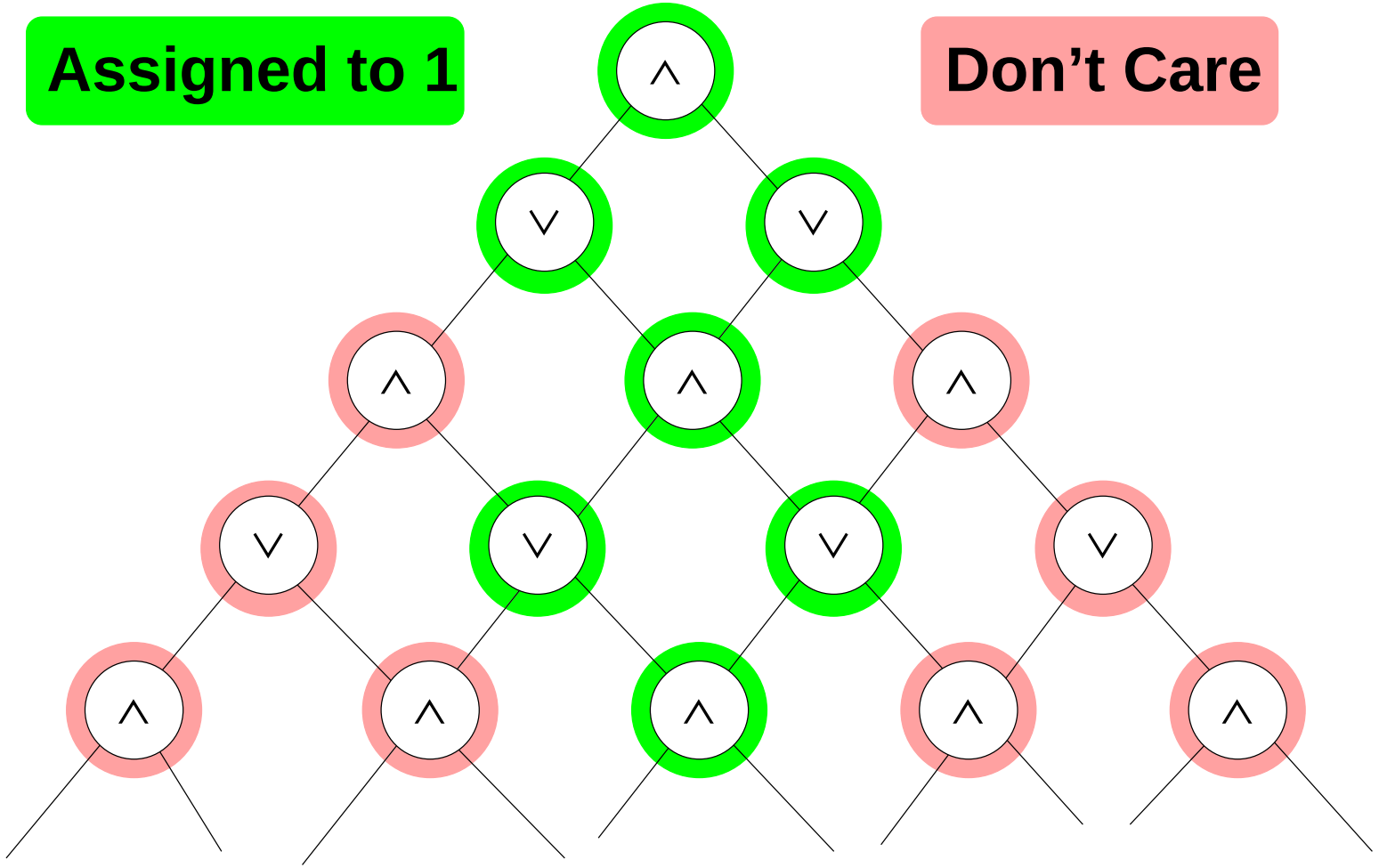
The Concept of Justification Frontiers



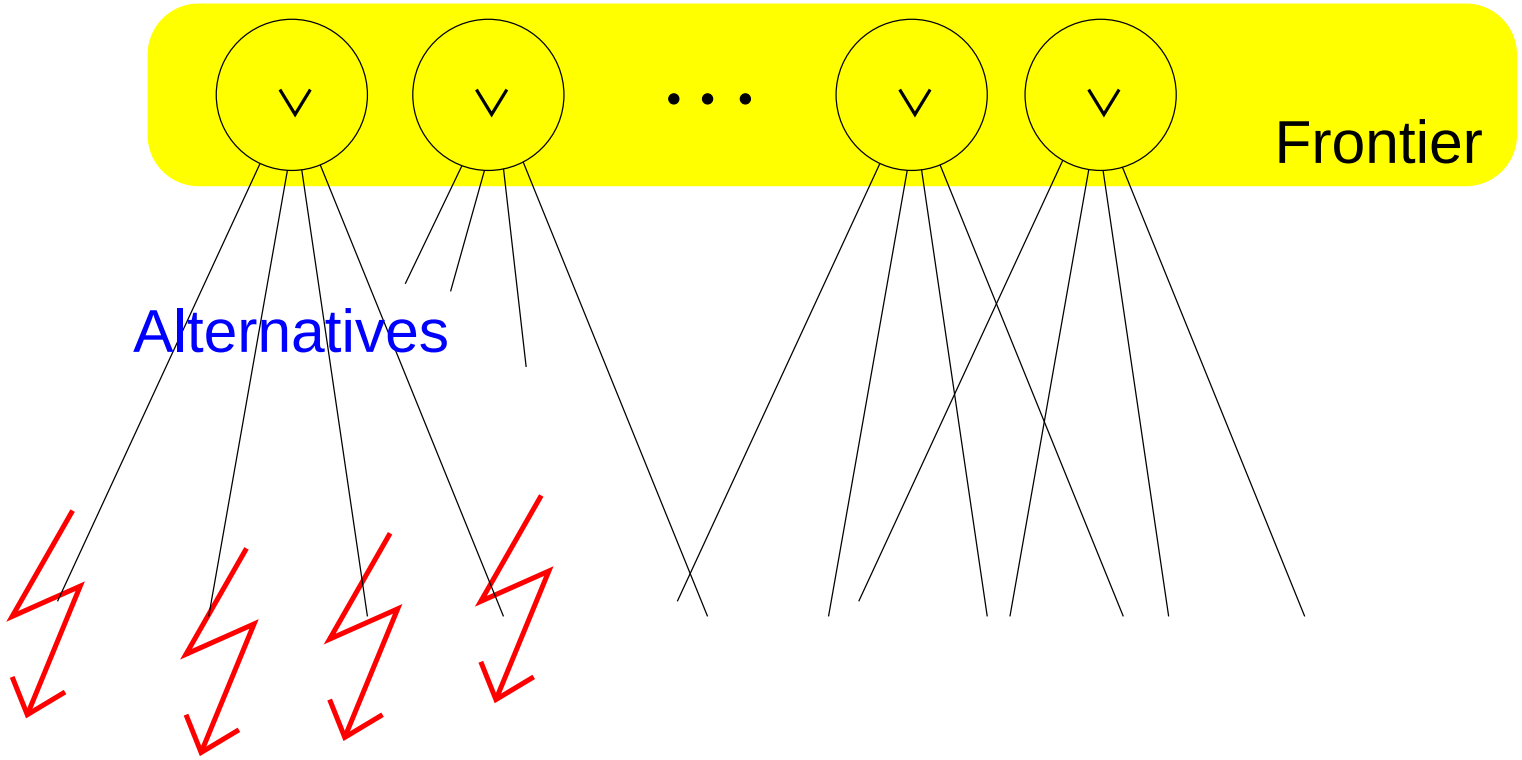
The Concept of Justification Frontiers



The Concept of Justification Frontiers



Local Contradictions in a Justification Procedure



If all local alternatives of one frontier element are exhausted
Then current global partial assignment is non valid

The Concept of Justification Frontiers

Reduces the search space by

(a) Searching for partial assignments only

(b) Trying to find local contradictions

Similar to Set-of-Support (SOS) and Model-Elimination in Resolution

Conjecture

Justification can be reformulated as resolution with an SOS strategy

Caching with a Clause Data Base

For completeness partial assignments are cached!

A partial assignment

$$\{x_1 \mapsto v_1, \ldots, x_n \mapsto v_n\} \quad \text{with } v_i \in \{0,1\}$$

is recorded as the following clause

$$y_1 \vee \ldots \vee y_n \quad \text{with } y_i = \begin{cases} x_i & \text{if } v_i = 0 \\ \neg x_i & \text{if } v_i = 1 \end{cases}$$

Check future assignments against the Clause Data Base!

Requirements for Clause Data Base

Datastructures for a Clause Data Base (CDB)

Check consistency on-the-fly

$\Rightarrow$ each single variable assignment implies a unit propagation

$\Rightarrow$ fast unit propagation

Each partial state assignment may generate a clause

$\Rightarrow$ fast addition of clauses

Promising Implementations: CNF, Trie, ...

Caching Justification Frontiers

Avoid checking the same frontiers again!

1. Dependency analysis on frontier elements while backtracking

2. Cache assumptions that induce the current frontier

$\Rightarrow$ frontier caching subsumes caching of partial state assignments

$\Rightarrow$ may record much larger clauses

$\Rightarrow$ often does not pay off (in our current implementation)

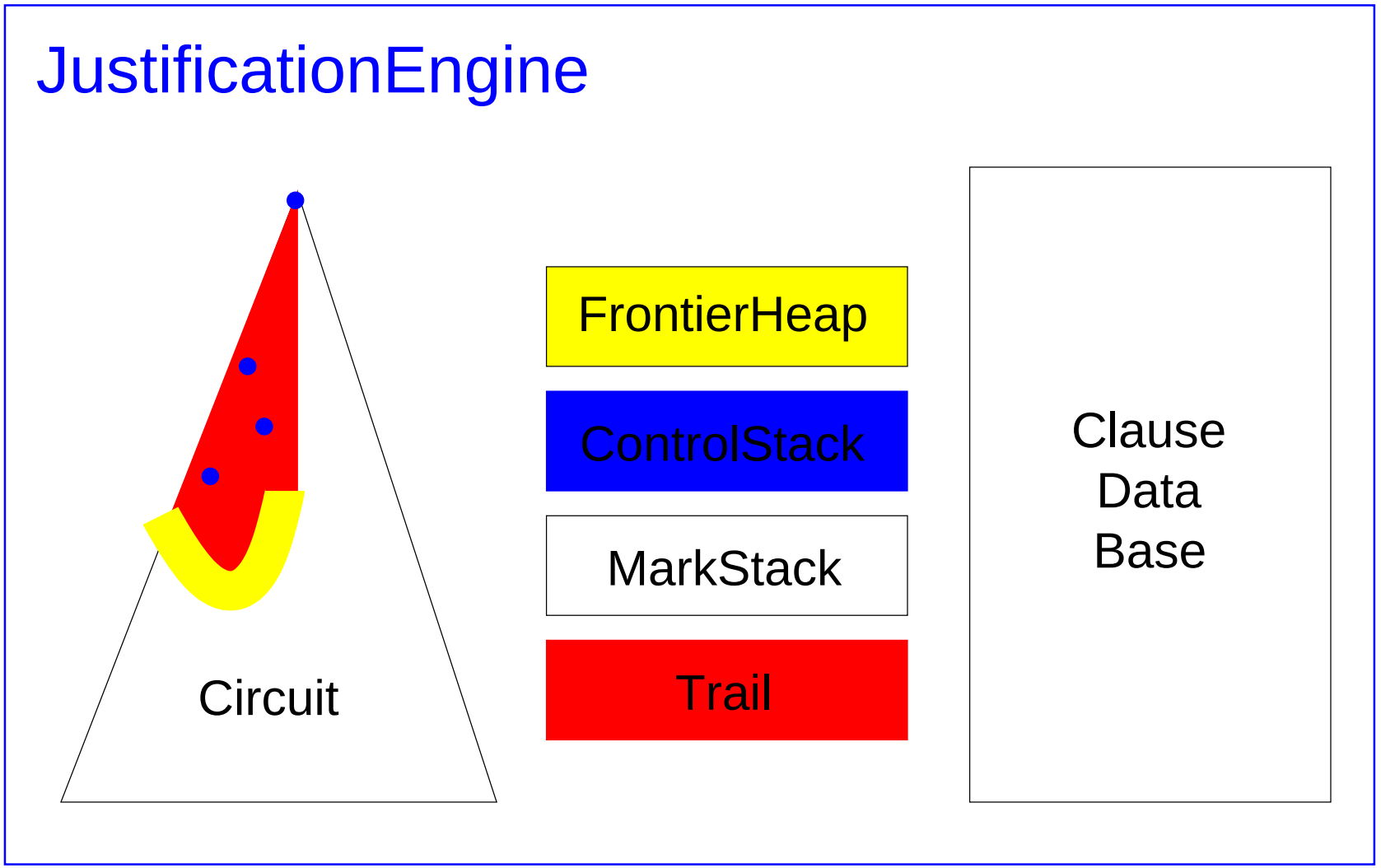
Justification Engine – Features

Features from ATPG

- Propagation of a Justification frontier
- SCOAP heuristic for ordering of frontier elements

Features from SAT

- Conflict directed backtracking
- Relevance learning
- Clause data base (CNF, Trie) for recording partial assignments



Why do we want general SMV Models?

Formulate boolean restrictions to states and transitions:

`INIT`, `INVAR`, `TRANS`

Interleaving semantics with `process` keyword

Comparison with model checkers and a wide range of applications!

Extension to other properties than just pure safety!

Other application domains beside EDA: protocols, optimization, ...

SMVFlatten	Hierarchical Flattening Boolean Encoding Elimination of Nondeterminism
FlatSMV	Fast Parser Backannotation Capability Trace Generator
CBMC	COI Reduction CNF generation Sequential Justification Engine Bounded Model Checker

[Boppana...] ATPG tools for model checking

[Biere...] [Sheeran...] Bounded Model Checking

[Gupta...] Using SAT procedures for faster Image calculation

[Een...] [Williams...] Image calculation by substitution

[Stoffel...] Structural state space search

[Plaisted...] CNF based QBF decision algorithm for image calculation

Conclusion

❶ Sequential Justification procedure ✔

❷ Caching of Partial State Assignments with a CDB ✔

❸ Caching of Justification Frontiers with a CDB ✔

❹ Non BDD based Synthesis of SMV Models ✔

❺ Several SAT based Model Checking Engines ✔

❻ Tool Chain for Model Checking and SAT experiments ✔

Future Work

① Application to realistic examples ?

② Experimental Comparison with related work ?

③ CDB implemented as Trie, BDD or ZBDD ?

④ Frame transition before frontier reaches state variables ?

⑤ Boolean constraint propagation in CDB only ?

⑥ VHDL and Verilog backend for SMVFlatten ?