CS181u Applied Logic & Automated Reasoning

Lecture 03: DPLL, Model Counting

Professor Lucas Bang
Harvey Mudd College
Department of Computer Science


NO PARKING
2AM TO 5AM
NIGHTLY
NO STOPPING
7AM TO 9AM
4PM TO 6PM
EXCEPT SATURDAY & SUNDAY
2 HOUR PARKING
SATURDAY
9AM TO 4PM
6PM TO 8PM
8AM TO 8PM
TOW-AWAY
TEMPORARY
NO PARKING
7AM TO 7PM
SUNDAY ONLY
TOW-AWAY
TEMPORARY
NO STOPPING
11AM TO MIDNIGHT
SATURDAY ONLY

Satisfiability for Boolean Logic

Given a formula ϕ, is it possible to assign all variables the values T or F so that the formula evaluates to T?

$$\phi = (x \vee y) \wedge (\neg x \vee z) \wedge (z \vee w) \wedge x \wedge (y \vee v)$$

Satisfiability for Boolean Logic

Given a formula ϕ, is it possible to assign all variables the values T or F so that the formula evaluates to T?

$$\phi = (x \vee y) \wedge (\neg x \vee z) \wedge (z \vee w) \wedge x \wedge (y \vee v)$$

$$(x, y, z, w, v) = (T, F, T, F, T)$$

Satisfiability for Boolean Logic

Given a formula ϕ, is it possible to assign all variables the values T or F so that the formula evaluates to T?

$$\phi = (x \vee y) \wedge (\neg x \vee z) \wedge (z \vee w) \wedge x \wedge (y \vee v)$$

$$(x, y, z, w, v) = (T, F, T, F, T)$$

A satisfying assignment is called a model for ϕ.

Satisfiability for Boolean Logic

Given a formula ϕ, is it possible to assign all variables the values T or F so that the formula evaluates to T?

$$\phi = (x \vee y) \wedge (\neg x \vee z) \wedge (z \vee w) \wedge x \wedge (y \vee v)$$

$$(x, y, z, w, v) = (T, F, T, F, T)$$

A satisfying assignment is called a model for ϕ.

A harder problem: *how many models are there?*

Model Counting for Boolean Logic

x	y	z	w	v	ϕ
F	F	F	F	F	F
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
T	F	F	T	T	F
T	F	T	F	F	F
T	F	T	F	T	T
T	F	T	T	F	F
T	F	T	T	T	T
T	T	F	F	F	F
T	T	F	F	T	F
T	T	F	T	F	F
T	T	F	T	T	F
T	T	T	F	F	T
T	T	T	F	T	T
T	T	T	T	F	T
T	T	T	T	T	T

Easy! Just make the truth table and count!

Model Counting for Boolean Logic

x	y	z	w	v	ϕ
F	F	F	F	F	F
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
T	F	F	T	T	F
T	F	T	F	F	F
T	F	T	F	T	T
T	F	T	T	F	F
T	F	T	T	T	T
T	T	F	F	F	F
T	T	F	F	T	F
T	T	F	T	F	F
T	T	F	T	T	F
T	T	T	F	F	T
T	T	T	F	T	T
T	T	T	T	F	T
T	T	T	T	T	T

Easy! Just make the truth table and count!

Model Counting for Boolean Logic

x	y	z	w	v	ϕ
F	F	F	F	F	F
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
T	F	F	T	T	F
T	F	T	F	F	F
T	F	T	F	T	T
T	F	T	T	F	F
T	F	T	T	T	T
T	T	F	F	F	F
T	T	F	F	T	F
T	T	F	T	F	F
T	T	F	T	T	F
T	T	T	F	F	T
T	T	T	F	T	T
T	T	T	T	F	T
T	T	T	T	T	T

Easy! Just make the truth table and count!

ϕ has 6 models.

Model Counting for Boolean Logic

x	y	z	w	v	ϕ
F	F	F	F	F	F
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
T	F	F	T	T	F
T	F	T	F	F	F
T	F	T	F	T	T
T	F	T	T	F	F
T	F	T	T	T	T
T	T	F	F	F	F
T	T	F	F	T	F
T	T	F	T	F	F
T	T	F	T	T	F
T	T	T	F	F	T
T	T	T	F	T	T
T	T	T	T	F	T
T	T	T	T	T	T

Easy! Just make the truth table and count!

ϕ has 6 models.

This approach is $\Theta(2^n)$.

Bummer!

Model Counting for Boolean Logic

In 1962, Davis, Putnam, Logemann, Loveland published the DPLL algorithm for Boolean SAT.

DPLL is the backbone of modern industry-grade automated theorem proving (Amazon, Microsoft, NASA, ...)

DPLL is also a model counting algorithm!

DPLL Execution Example

$$z \wedge x \wedge (y \vee v)$$

$$\{z, x, y \vee v\}$$

DPLL Execution Example

$$\{z, x, y \vee v\}$$

$$x \mapsto F$$

UNSAT $\{z, F, y \vee v\}$

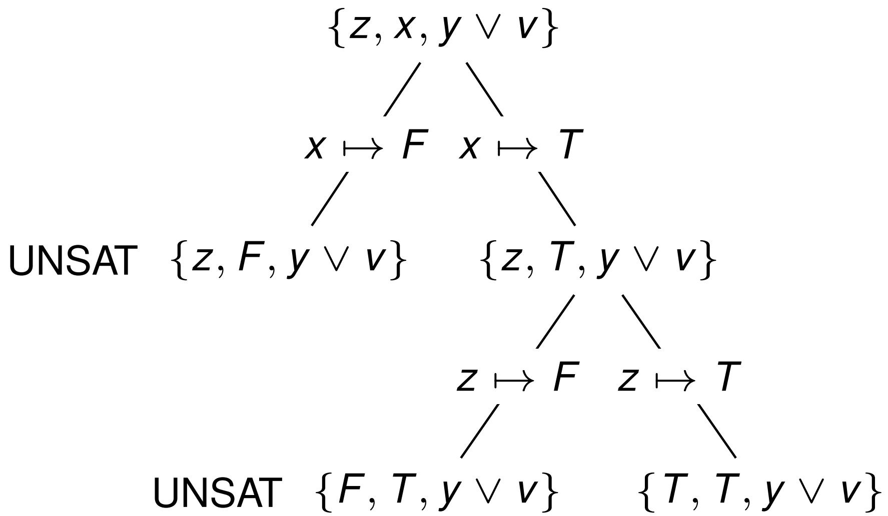
DPLL Execution Example

$$
\begin{array}{ccc}
 & \{z, x, y \vee v\} & \\
 & x \mapsto F \qquad x \mapsto T & \\
\text{UNSAT} \quad \{z, F, y \vee v\} & & \{z, T, y \vee v\}
\end{array}
$$

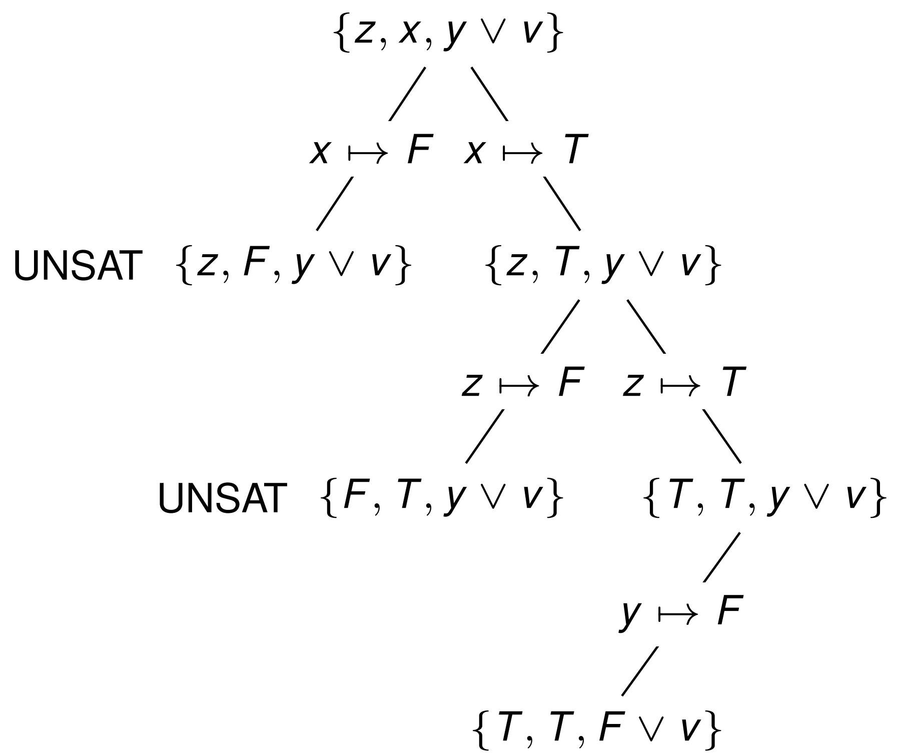
DPLL Execution Example

$$
\begin{tikzcd}[column sep=small]
 & & & \{z, x, y \vee v\} \arrow[dl, dash] \arrow[dr, dash] & & \\
 & & x \mapsto F \arrow[dl, dash] & & x \mapsto T \arrow[dr, dash] & \\
\text{UNSAT} & \{z, F, y \vee v\} & & & & \{z, T, y \vee v\} \arrow[dl, dash] \\
 & & & & z \mapsto F \arrow[dl, dash] & \\
 & & \text{UNSAT} & \{F, T, y \vee v\} & &
\end{tikzcd}
$$

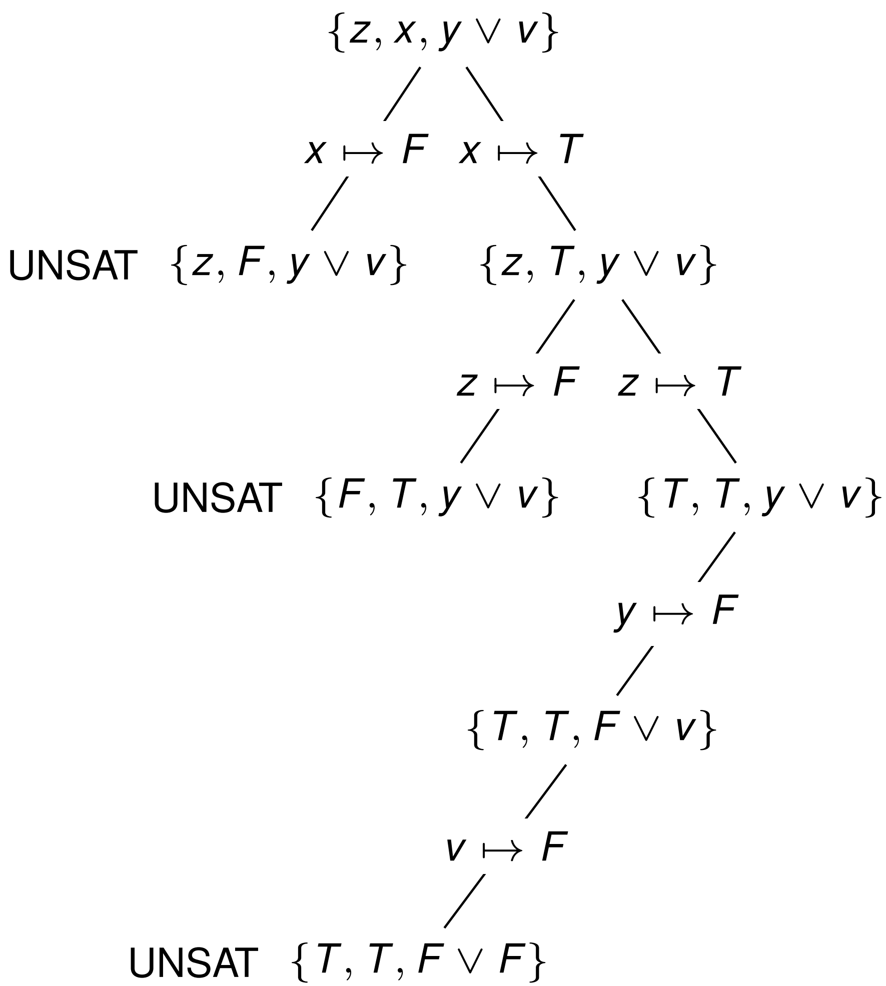
DPLL Execution Example



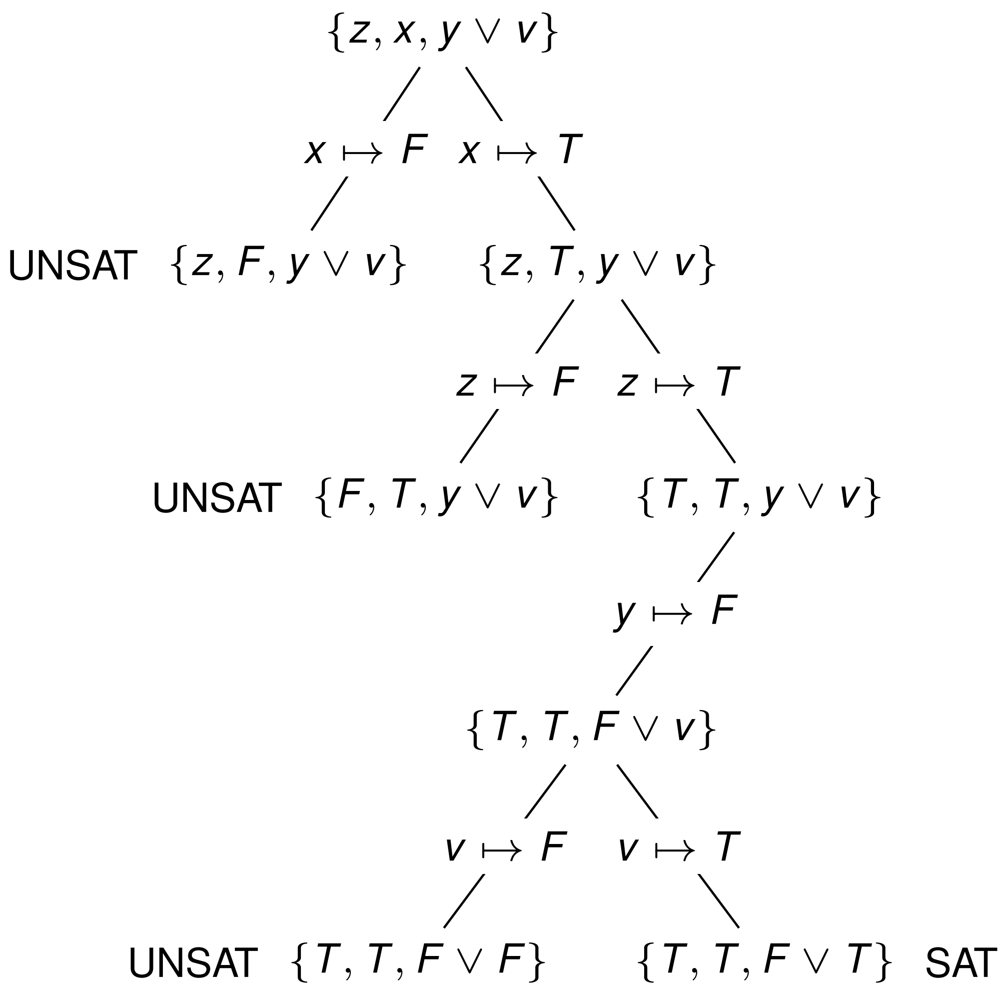
DPLL Execution Example



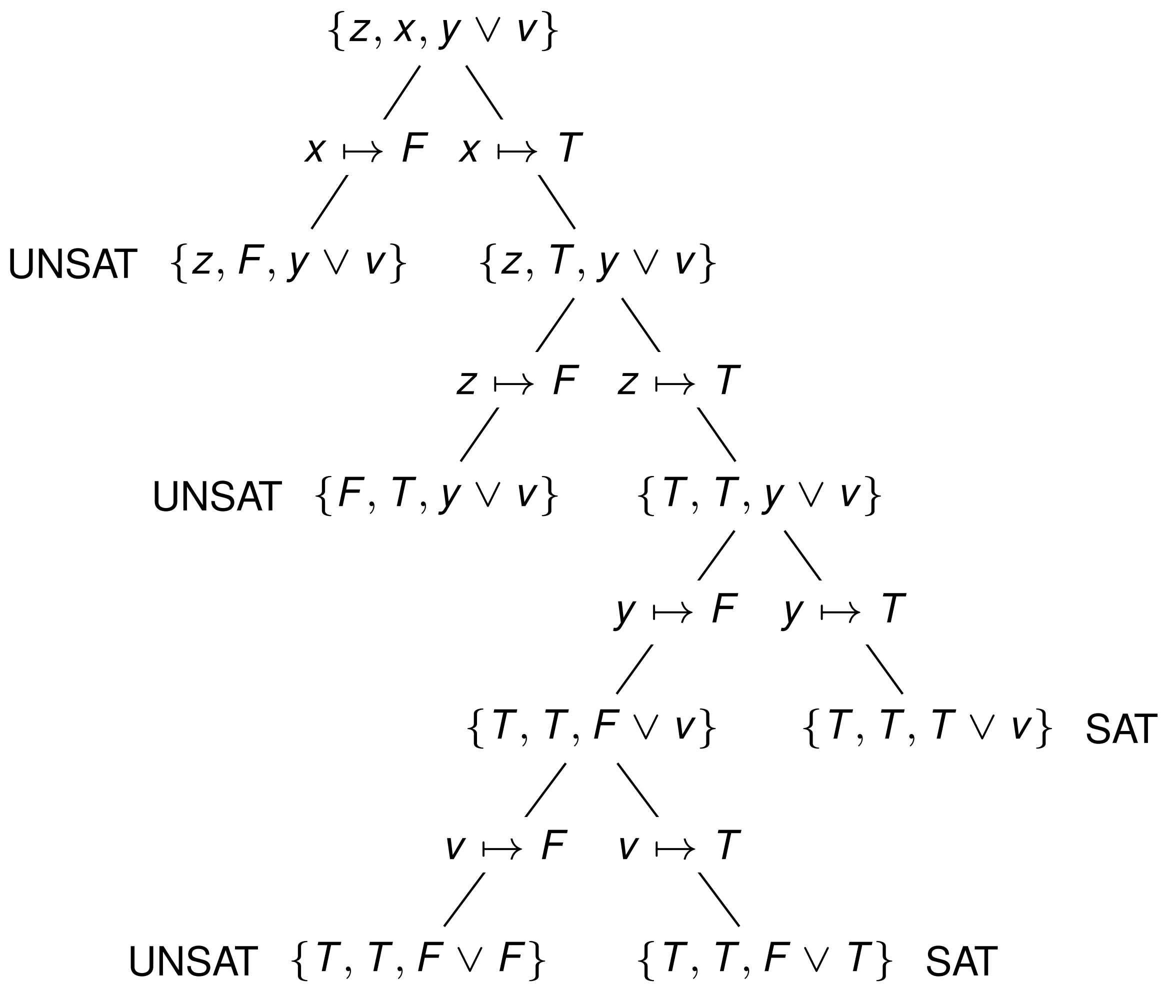
DPLL Execution Example



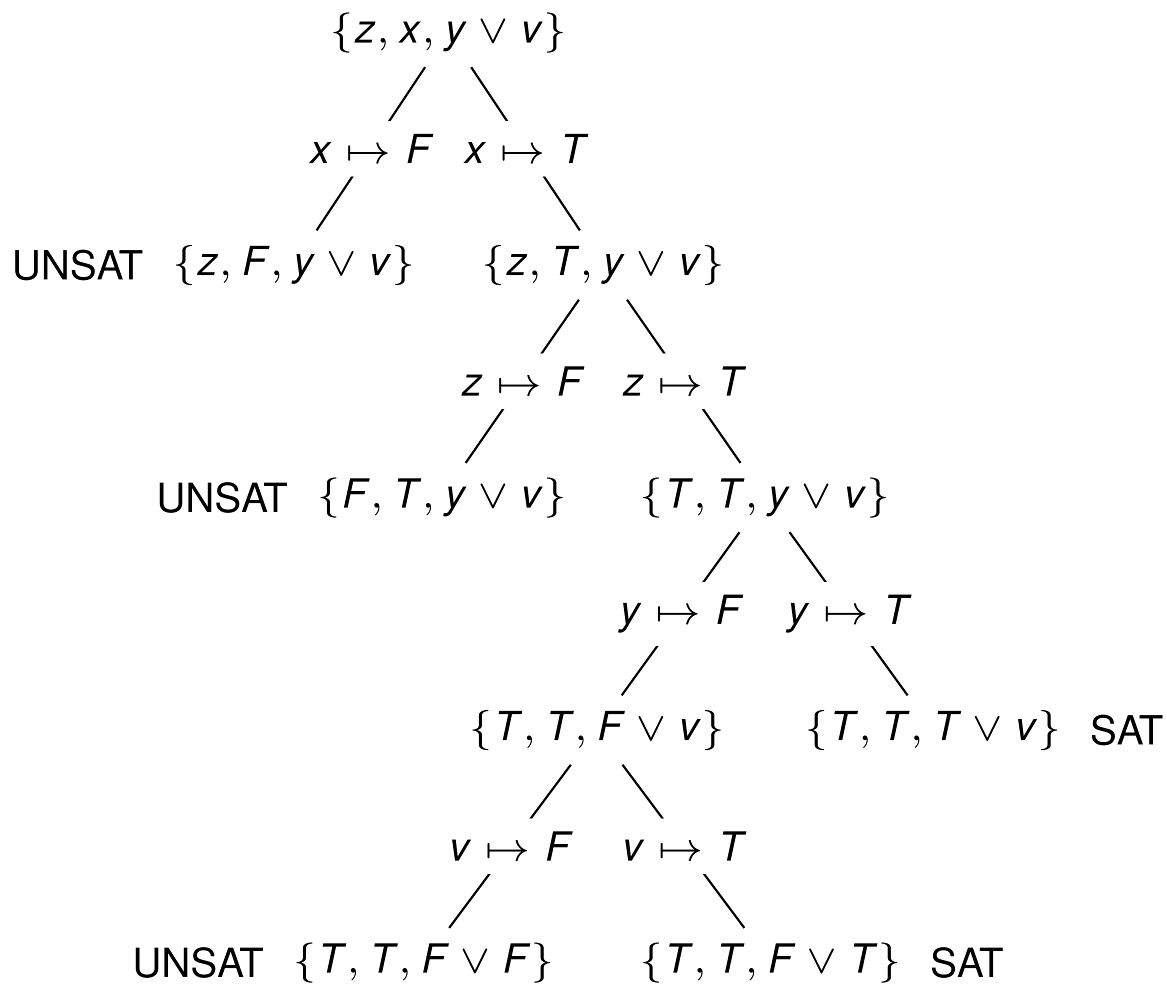
DPLL Execution Example



DPLL Execution Example



DPLL Execution Example



Conclusion: ϕ is satisfiable

Davis-Putnam-Logemann-Loveland (DPLL) Algorithm

Function : DPLL(ϕ)
Input : CNF formula ϕ over *n* variables
Output : true or false, the satisfiability of F
begin
  UnitPropagate(ϕ)
  if ϕ has false clause **then return** false
  if all clauses of ϕ satisfied **then return** true
  x $\leftarrow$ SelectBranchVariable(ϕ)
  return DPLL($\phi[x \mapsto \mathit{true}]$) $\vee$ DPLL($\phi[x \mapsto \mathit{false}]$)
end

Davis-Putnam-Logemann-Loveland (DPLL) Algorithm

Function : DPLL(ϕ)
Input : CNF formula ϕ over n variables
Output : true or false, the satisfiability of F
begin
  UnitPropagate(ϕ)
  if ϕ has false clause **then return** false
  if all clauses of ϕ satisfied **then return** true
  x $\leftarrow$ SelectBranchVariable(ϕ)
  return DPLL($\phi[x \mapsto \textit{true}]$) $\vee$ DPLL($\phi[x \mapsto \textit{false}]$)
end

Davis-Putnam-Logemann-Loveland (DPLL) Algorithm

Function : DPLL(ϕ)
Input : CNF formula ϕ over *n* variables
Output : true or false, the satisfiability of F
begin
  UnitPropagate(ϕ)
  if ϕ has false clause **then return** false
  if all clauses of ϕ satisfied **then return** true
  x $\leftarrow$ SelectBranchVariable(ϕ)
  return DPLL($\phi[x \mapsto \mathit{true}]$) $\vee$ DPLL($\phi[x \mapsto \mathit{false}]$)
end

Davis-Putnam-Logemann-Loveland (DPLL) Algorithm

Function : DPLL(ϕ)
Input : CNF formula ϕ over *n* variables
Output : true or false, the satisfiability of F
begin
  UnitPropagate(ϕ)
  if ϕ has false clause **then return** false
  if all clauses of ϕ satisfied **then return** true
  $\boxed{\text{x} \leftarrow \text{SelectBranchVariable}(\phi)}$
  return DPLL($\phi[x \mapsto \textit{true}]$) $\vee$ DPLL($\phi[x \mapsto \textit{false}]$)
end

Davis-Putnam-Logemann-Loveland (DPLL) Algorithm

Function : DPLL(ϕ)
Input : CNF formula ϕ over *n* variables
Output : true or false, the satisfiability of F
begin
  UnitPropagate(ϕ)
  if ϕ has false clause **then return** false
  if all clauses of ϕ satisfied **then return** true
  x $\leftarrow$ SelectBranchVariable(ϕ)
  return DPLL($\phi[x \mapsto \mathit{true}]$) $\vee$ DPLL($\phi[x \mapsto \mathit{false}]$)
end

DPLL Execution Example

$$z \wedge x \wedge (y \vee v)$$

$t = 4$ $\{z, x, y \vee v\}$

DPLL Execution Example

$t = 4$ $\{z, x, y \vee v\}$

$x \mapsto F$

$t = 3$ UNSAT $\{z, F, y \vee v\}$

DPLL Execution Example

$t = 4$ $\{z, x, y \vee v\}$

$x \mapsto F$ $x \mapsto T$

$t = 3$ UNSAT $\{z, F, y \vee v\}$ $\{z, T, y \vee v\}$ $t = 3$

DPLL Execution Example

$t = 4$ $\{z, x, y \vee v\}$

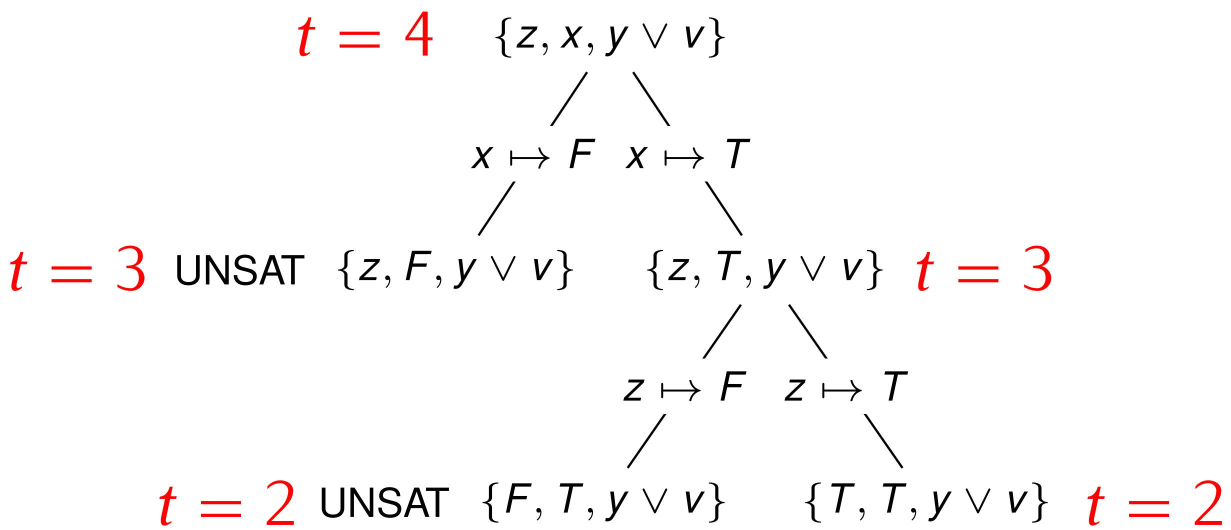
$x \mapsto F$ $x \mapsto T$

$t = 3$ UNSAT $\{z, F, y \vee v\}$ $\{z, T, y \vee v\}$ $t = 3$

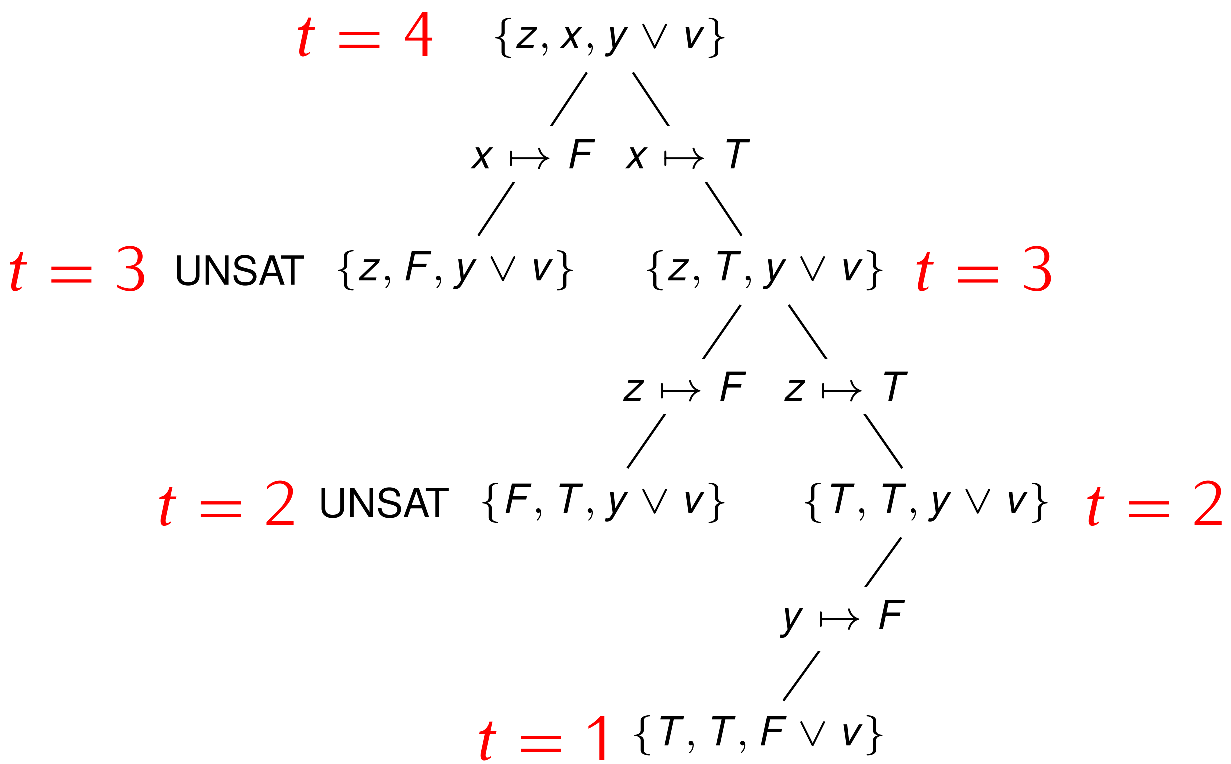
$z \mapsto F$

$t = 2$ UNSAT $\{F, T, y \vee v\}$

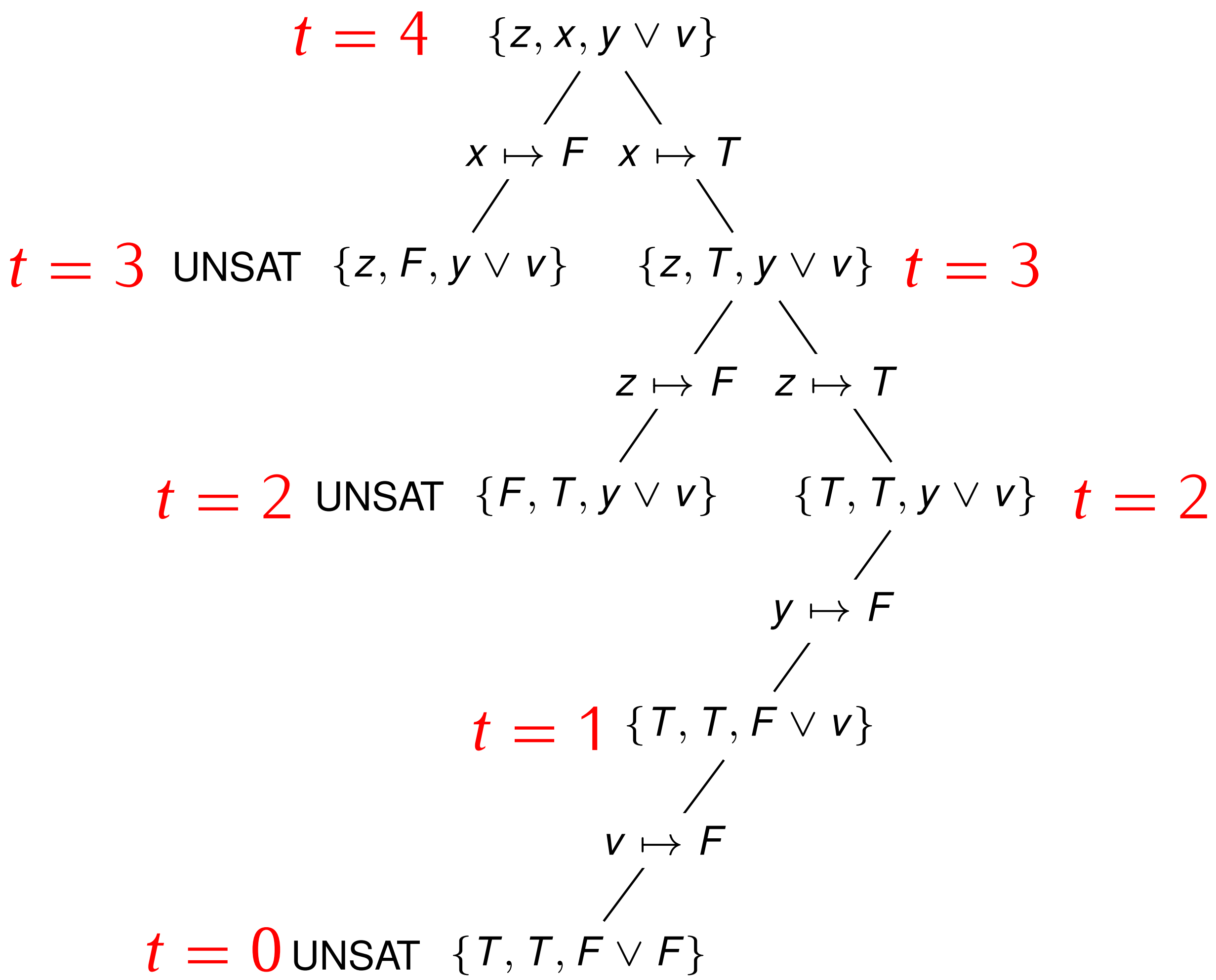
DPLL Execution Example



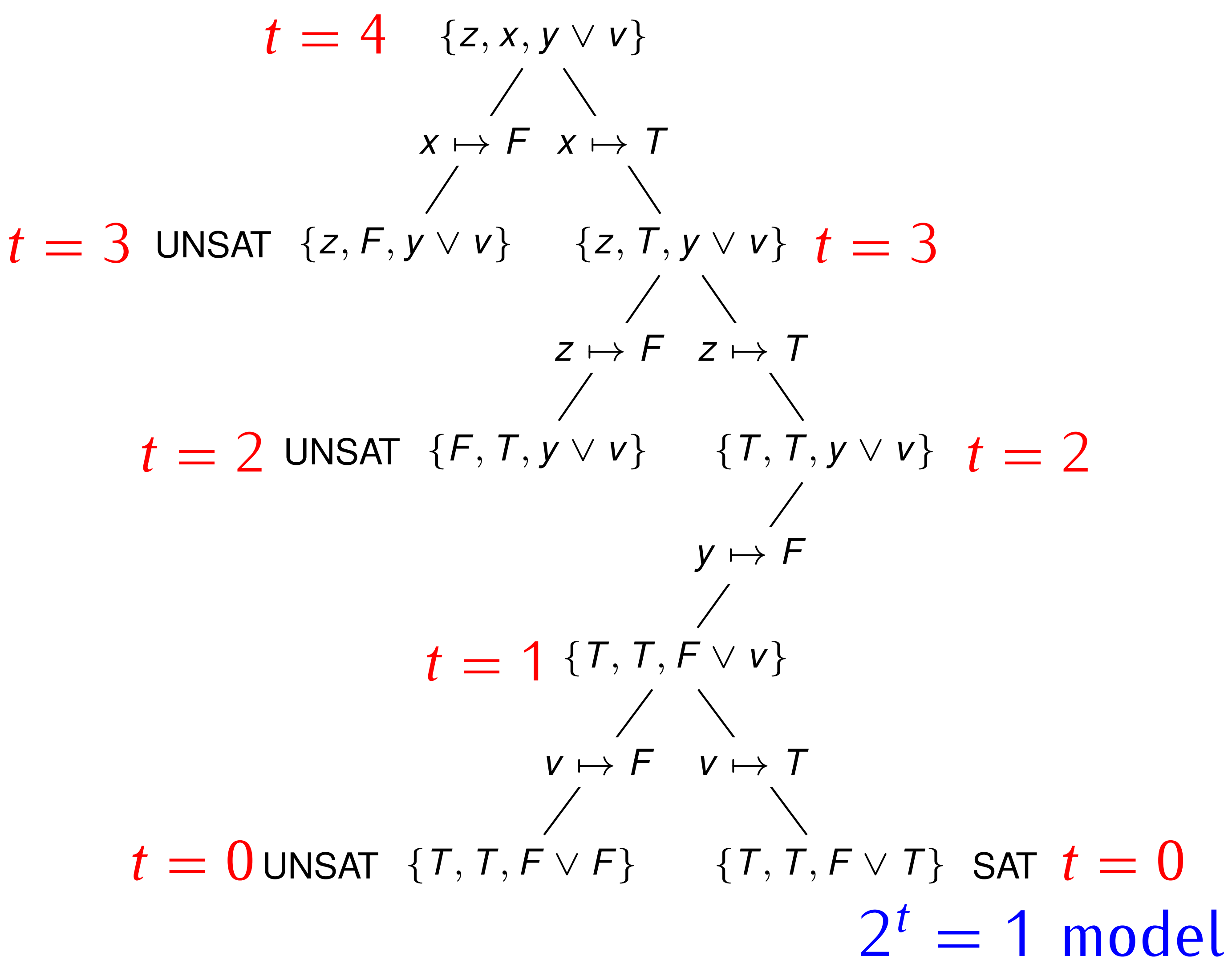
DPLL Execution Example



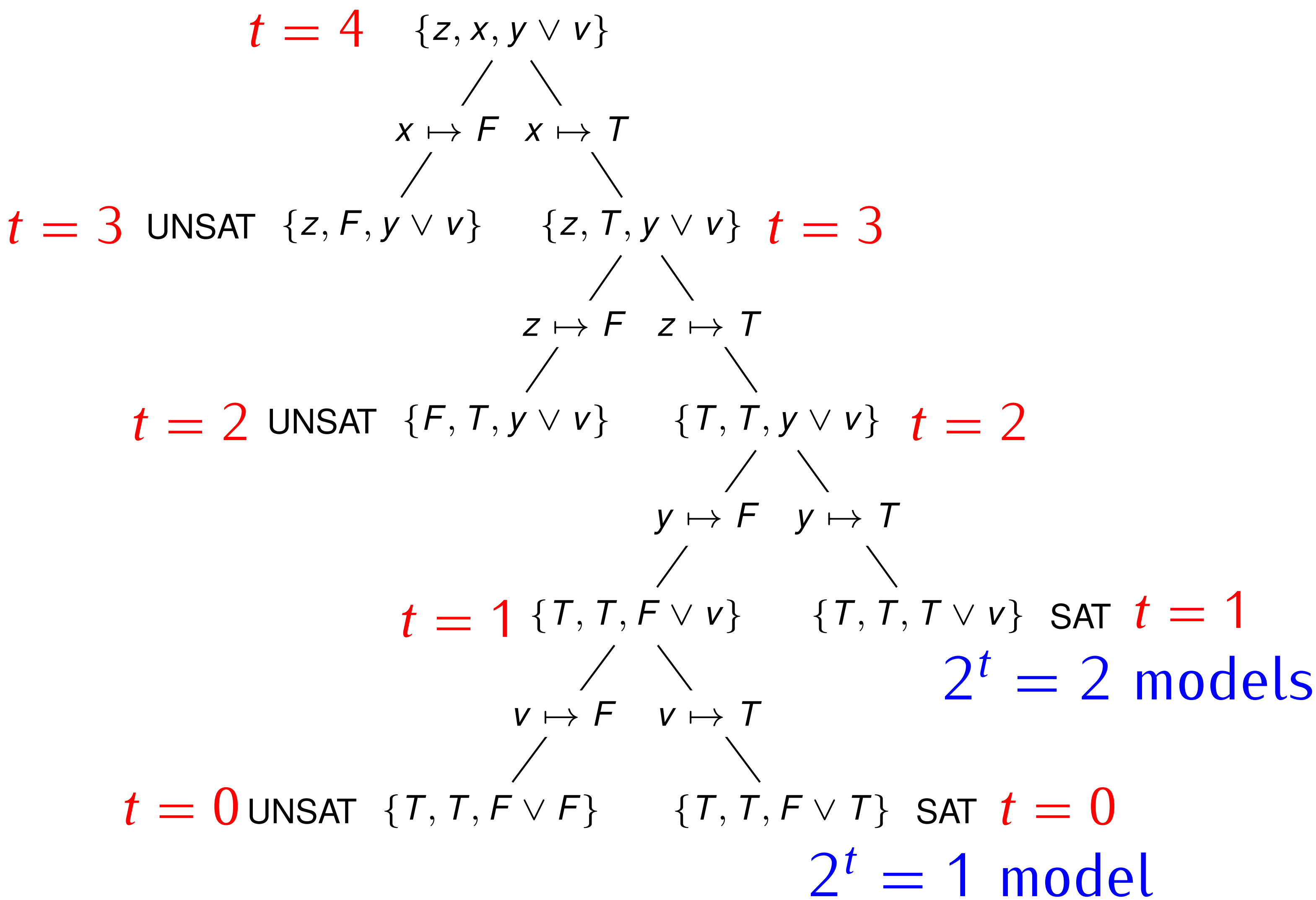
DPLL Execution Example



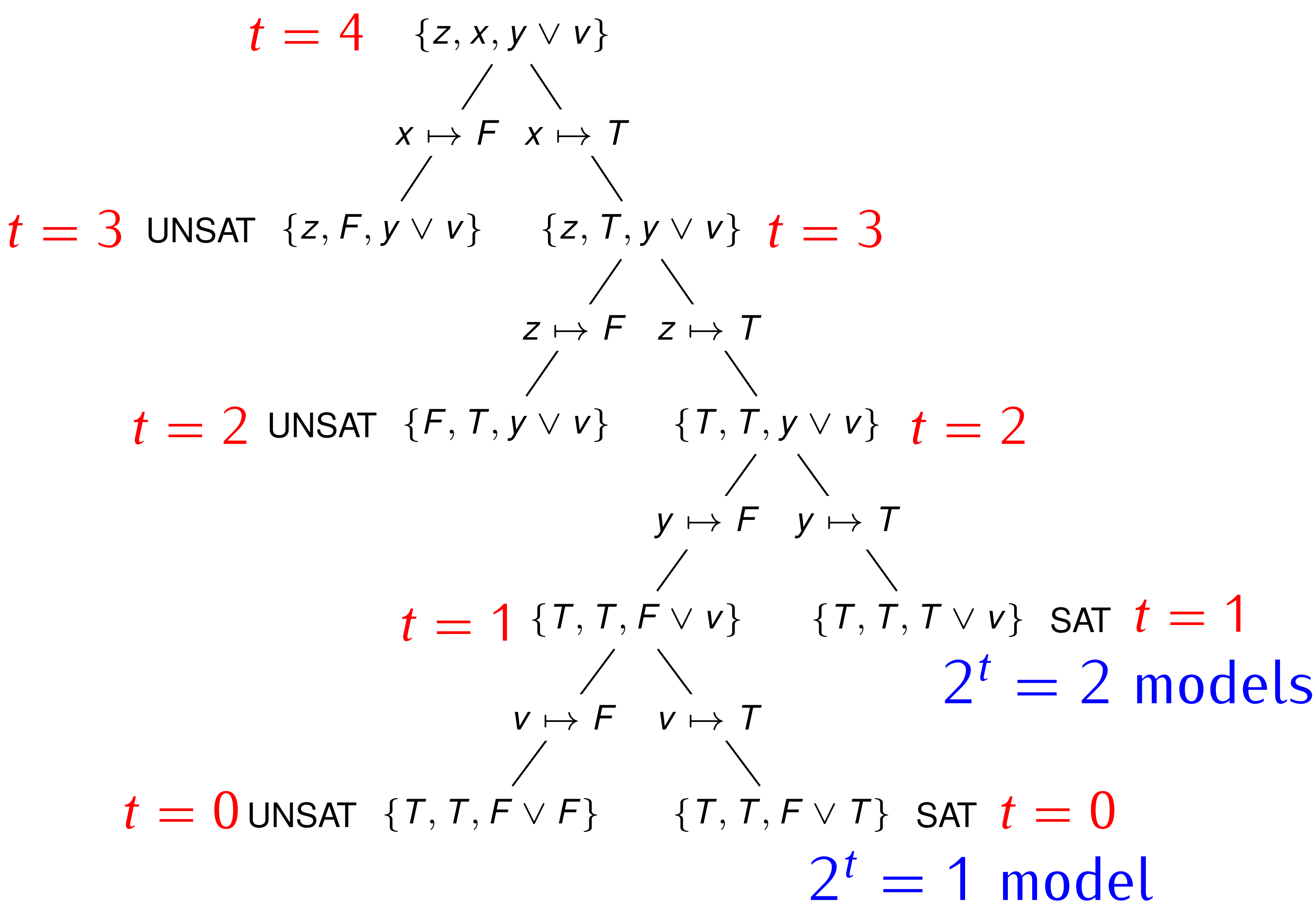
DPLL Execution Example



DPLL Execution Example



DPLL Execution Example



Conclusion: ϕ has 3 models

Davis-Putnam-Logemann-Loveland (DPLL) Algorithm

DPLL can be converted into a procedure for $\#$CNF-SAT.

Function : DPLL(ϕ, t)
Input : CNF formula ϕ over *n* variables; $t \in \mathbb{Z}$
Output : $\#\phi$, the model count of ϕ
begin
  UnitPropagate(ϕ)
  if ϕ has false clause **then return** *false*
  if all clauses of ϕ satisfied **then return** *true*
  x $\leftarrow$ SelectBranchVariable(ϕ)
  return DPLL(ϕ[*x* $\mapsto$ *true*], $t - 1$) $\vee$ DPLL(ϕ[*x* $\mapsto$ *true*], $t - 1$)
end

Davis-Putnam-Logemann-Loveland (DPLL) Algorithm

DPLL can be converted into a procedure for $\#$CNF-SAT.

Function : DPLL(ϕ, $\boxed{t}$)
Input : CNF formula ϕ over n variables; $\boxed{t \in \mathbb{Z}}$
Output : $\#\phi$, the model count of ϕ
begin
    UnitPropagate(ϕ)
    if ϕ has false clause **then return** *false*
    if all clauses of ϕ satisfied **then return** *true*
    x $\leftarrow$ SelectBranchVariable(ϕ)
    return DPLL($\phi[x \mapsto \mathit{true}]$, $\boxed{t-1}$) $\vee$ DPLL($\phi[x \mapsto \mathit{true}]$, $\boxed{t-1}$)
end

Davis-Putnam-Logemann-Loveland (DPLL) Algorithm

DPLL can be converted into a procedure for $\#$CNF-SAT.

Function : DPLL(ϕ, t)
Input : CNF formula ϕ over *n* variables; $t \in \mathbb{Z}$
Output : $\#\phi$, the model count of ϕ
begin
  UnitPropagate(ϕ)
  if ϕ has false clause **then return** $\boxed{\textit{false}}$
  if all clauses of ϕ satisfied **then return** *true*
  x $\leftarrow$ SelectBranchVariable(ϕ)
  return DPLL($\phi[x \mapsto \textit{true}], t-1$) $\vee$ DPLL($\phi[x \mapsto \textit{true}], t-1$)
end

Davis-Putnam-Logemann-Loveland (DPLL) Algorithm

DPLL can be converted into a procedure for $\#$CNF-SAT.

Function : DPLL(ϕ, t)
Input : CNF formula ϕ over n variables; $t \in \mathbb{Z}$
Output : $\#\phi$, the model count of ϕ
begin
  UnitPropagate(ϕ)
  if ϕ has false clause **then return** $\boxed{0}$
  if all clauses of ϕ satisfied **then return** *true*
  x $\leftarrow$ SelectBranchVariable(ϕ)
  return DPLL($\phi[x \mapsto \mathit{true}], t-1$) $\vee$ DPLL($\phi[x \mapsto \mathit{true}], t-1$)
end

Davis-Putnam-Logemann-Loveland (DPLL) Algorithm

DPLL can be converted into a procedure for $\#$CNF-SAT.

Function : DPLL(ϕ, t)
Input : CNF formula ϕ over n variables; $t \in \mathbb{Z}$
Output : $\#\phi$, the model count of ϕ
begin
    UnitPropagate(ϕ)
    if ϕ has false clause **then return** 0
    if all clauses of ϕ satisfied **then return** $\boxed{\textit{true}}$
    x $\leftarrow$ SelectBranchVariable(ϕ)
    return DPLL($\phi[x \mapsto \textit{true}], t-1$) $\vee$ DPLL($\phi[x \mapsto \textit{true}], t-1$)
end

Davis-Putnam-Logemann-Loveland (DPLL) Algorithm

DPLL can be converted into a procedure for $\#$CNF-SAT.

Function : DPLL(ϕ, t)
Input : CNF formula ϕ over n variables; $t \in \mathbb{Z}$
Output : $\#\phi$, the model count of ϕ
begin
UnitPropagate(ϕ)
if ϕ has false clause **then return** 0
if all clauses of ϕ satisfied **then return** $\boxed{2^t}$
x $\leftarrow$ SelectBranchVariable(ϕ)
return DPLL($\phi[x \mapsto \mathit{true}], t-1) \vee$ DPLL($\phi[x \mapsto \mathit{true}], t-1$)
end

Davis-Putnam-Logemann-Loveland (DPLL) Algorithm

DPLL can be converted into a procedure for $\#$CNF-SAT.

Function : DPLL(ϕ, t)
Input : CNF formula ϕ over n variables; $t \in \mathbb{Z}$
Output : $\#\phi$, the model count of ϕ
begin
  UnitPropagate(ϕ)
  if ϕ has false clause **then return** 0
  if all clauses of ϕ satisfied **then return** 2^t
  x $\leftarrow$ SelectBranchVariable(ϕ)
  return DPLL($\phi[x \mapsto \mathit{true}], t-1$) $\boxed{+}$ DPLL($\phi[x \mapsto \mathit{true}], t-1$)
end

The Big Idea

DPLL $\longrightarrow$ #DPLL

Satifiability Algorithm $\longrightarrow$ Counting Algorithm

Model Counting for Arrays

$$\forall i : 0 \leq a[i] < k \wedge 0 \leq b[i] < k$$

$$\forall i : a[i] \neq b[i]$$

$$\text{length}(a) = n$$

$$\text{length}(b) = n$$

Model Counting for Arrays

$$\forall i : 0 \leq a[i] < k \wedge 0 \leq b[i] < k$$

$$\forall i : a[i] \neq b[i]$$

$$\text{length}(a) = n$$

$$\text{length}(b) = n$$

First, is this SAT?

$k = 4, n = 5, a = [1, 1, 1, 1, 1], b = [0, 2, 3, 2, 0]$

Model Counting for Arrays

$$\forall i : 0 \leq a[i] < k \wedge 0 \leq b[i] < k$$

$$\forall i : a[i] \neq b[i]$$

$$\text{length}(a) = n$$

$$\text{length}(b) = n$$

First, is this SAT?

$k = 4, n = 5, a = [1, 1, 1, 1, 1], b = [0, 2, 3, 2, 0]$

For a given k and n, how many models?

$$\#\phi(k, n) = (k^2 - k)^n$$

Model Counting for Arrays

$$\left.\begin{array}{l} \forall i : 0 \leq a[i] < k \\ \forall i, j : i < j \Rightarrow a[i] < a[j] \\ \text{length}(a) = n \\ \text{length}(b) = n \end{array}\right] \text{— } a \text{ is sorted}$$

Model Counting for Arrays

$$\left.\begin{array}{l}
\forall i : 0 \leq a[i] < k \\
\forall i : i < pivot \Rightarrow a[i] < a[pivot] \\
\forall i : i > pivot \Rightarrow a[i] > a[pivot] \\
\text{length}(a) = n
\end{array}\right] \text{ quicksort partitioning}$$

Model Counting for Arrays

$$\left.\begin{array}{l} \forall i : 0 \leq a[i] < k \\ \forall i : i < pivot \Rightarrow a[i] < a[pivot] \\ \forall i : i > pivot \Rightarrow a[i] > a[pivot] \\ \text{length}(a) = n \end{array}\right] \text{quicksort partitioning}$$

We could do ad-hoc analysis for every constraint, but we want an algorithm!

Model Counting for Arrays

The SAT algorithm
for arrays

$\phi \in$ Array Theory

Model Counting for Arrays

The SAT algorithm
for arrays

$$\begin{array}{l} \phi \in \text{Array Theory} \\ \Big\downarrow \; T_1 : A \rightarrow UIF \\ \phi' \in \text{UIF Theory} \end{array}$$

Model Counting for Arrays

The SAT algorithm
for arrays

$\phi \in$ Array Theory

$$\Big\downarrow \; T_1 : A \rightarrow UIF$$

$\phi' \in$ UIF Theory

$$\Big\downarrow \; T_2 : UIF \rightarrow EL$$

$\phi'' \in$ Equality Logic

Model Counting for Arrays

The SAT algorithm
for arrays

$$\phi \in \text{Array Theory}$$

$$\Big\downarrow \; T_1 : A \rightarrow UIF$$

$$\phi' \in \text{UIF Theory}$$

$$\Big\downarrow \; T_2 : UIF \rightarrow EL$$

$$\phi'' \in \text{Equality Logic}$$

There is a SAT
algorithm for this!

Model Counting for Arrays

The SAT algorithm for arrays	(Proposed) counting algorithm for arrays
$\phi \in$ Array Theory	$\phi \in$ Array Theory

$\phi \in$ Array Theory

$\downarrow$ $T_1 : A \rightarrow UIF$

$\phi' \in$ UIF Theory

$\downarrow$ $T_2 : UIF \rightarrow EL$

$\phi'' \in$ Equality Logic

There is a SAT algorithm for this!

Model Counting for Arrays

The SAT algorithm
for arrays

$\phi \in$ Array Theory

$$\Big\downarrow \; T_1 : A \rightarrow UIF$$

$\phi' \in$ UIF Theory

$$\Big\downarrow \; T_2 : UIF \rightarrow EL$$

$\phi'' \in$ Equality Logic

There is a SAT
algorithm for this!

(Proposed) counting
algorithm for arrays

$\phi \in$ Array Theory

$$\Big\downarrow \; T_1' : A \rightarrow UIF$$

$\phi' \in$ UIF Theory

Model Counting for Arrays

The SAT algorithm for arrays	(Proposed) counting algorithm for arrays
$\phi \in$ Array Theory	$\phi \in$ Array Theory
$\downarrow T_1 : A \to UIF$	$\downarrow T_1' : A \to UIF$
$\phi' \in$ UIF Theory	$\phi' \in$ UIF Theory
$\downarrow T_2 : UIF \to EL$	$\downarrow T_2' : UIF \to EL$
$\phi'' \in$ Equality Logic	$\phi'' \in$ Equality Logic

There is a SAT algorithm for this!

Model Counting for Arrays

The SAT algorithm
for arrays

$$\begin{array}{l} \phi \in \text{Array Theory} \\ \quad \downarrow \; T_1 : A \rightarrow UIF \\ \phi' \in \text{UIF Theory} \\ \quad \downarrow \; T_2 : UIF \rightarrow EL \\ \phi'' \in \underbrace{\text{Equality Logic}} \end{array}$$

There is a SAT
algorithm for this!

(Proposed) counting
algorithm for arrays

$$\begin{array}{l} \phi \in \text{Array Theory} \\ \quad \downarrow \; T_1' : A \rightarrow UIF \\ \phi' \in \text{UIF Theory} \\ \quad \downarrow \; T_2' : UIF \rightarrow EL \\ \phi'' \in \text{Equality Logic} \\ \quad \downarrow \; T_3' : EL \rightarrow LIA \\ \phi''' \in \text{Linear Integer Arithmetic} \end{array}$$

Model Counting for Arrays

The SAT algorithm for arrays

$\phi \in$ Array Theory

$\downarrow \; T_1 : A \rightarrow UIF$

$\phi' \in$ UIF Theory

$\downarrow \; T_2 : UIF \rightarrow EL$

$\phi'' \in$ Equality Logic

There is a SAT algorithm for this!

(Proposed) counting algorithm for arrays

$\phi \in$ Array Theory

$\downarrow \; T_1' : A \rightarrow UIF$

$\phi' \in$ UIF Theory

$\downarrow \; T_2' : UIF \rightarrow EL$

$\phi'' \in$ Equality Logic

$\downarrow \; T_3' : EL \rightarrow LIA$

$\phi''' \in$ Linear Integer Arithmetic

There is a poly-time counting algorithm for this!

Model Counting Overview

Satisfiability Algorithm $\longrightarrow$ Counting Algorithm

DPLL $\longrightarrow$ #DPLL

Model Counting Overview

Satisfiability Algorithm $\longrightarrow$ Counting Algorithm

DPLL $\longrightarrow$ #DPLL

$\mathrm{SAT}_{\mathrm{Arr}} \longrightarrow \#\mathrm{SAT}_{\mathrm{Arr}}$