Satisfiability modulo theories and particle physics

September 25, 2024

Bakar Chargeishvili

CRC Young Scientists Meeting 2024
Karlsruhe, September 25-27

Karlsruhe Institute of Technology

Bridging Logics and Complexity in Particle Physics

The Challenge

- Particle physics problems inherently incorporate a set of logical operations
- Tracking logics becomes difficult as problem complexity increases
- High-precision phenomenological calculations are increasingly important

The Growing Complexity

- High number of literals in calculations:
  - Numerous propagators
  - Multiple scales
  - Abundant amplitudes

Bridging Logics and Complexity in Particle Physics

Current Limitations

- Traditional methods are slowly hitting a wall
- Often only relying on increased computing resources:
  - Results in longer waiting times
  - Increases CO2 footprint
  - Might not be optimal use of a human and computer time

A Promising Solution (for some problems)

- SMT (Satisfiability Modulo Theories) solvers
- Potential to optimize how we handle complex logics in particle physics

This Talk

We'll explore how SMT solvers can help us solve certain class of particle physics problems more efficiently and effectively.

Outline

Understanding SMT Solvers

What is SMT?

- SMT: Satisfiability Modulo Theories
- A decision problem for logical formulas with respect to combinations of background theories expressed in classical first-order logic

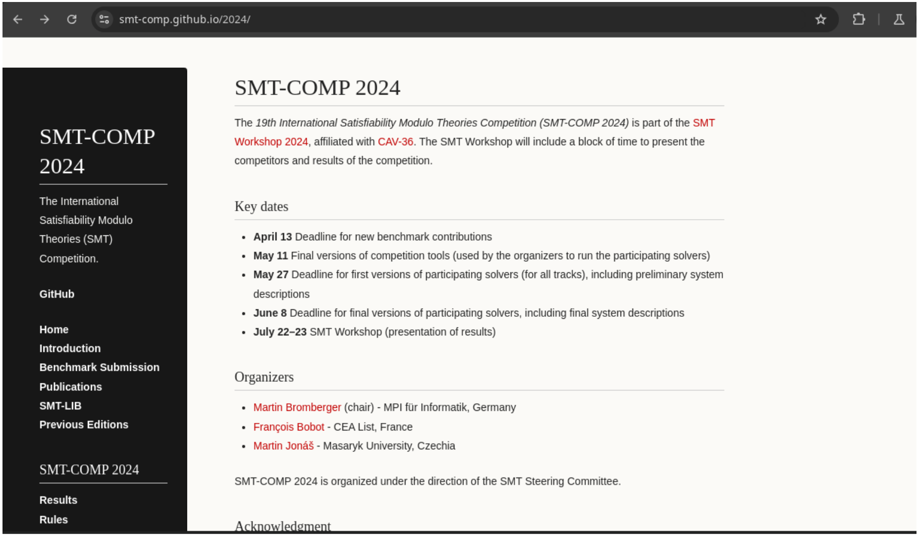
Historical Context

- Emerged in the late 1970s and early 1980s
- Gained significant traction in the 2000s with the advent of efficient SMT solvers

Famous SMT Solvers

- Z3 (Microsoft Research)
- CVC4/CVC5 (Stanford University)
- Yices (SRI International)
- MathSAT (FBK-IRST and University of Trento)
- SMT-RAT (RWTH Aachen University)

Competitive and rapidly growing area of research



SMT-COMP 2024

The International Satisfiability Modulo Theories (SMT) Competition.

SMT-COMP 2024

The *19th International Satisfiability Modulo Theories Competition (SMT-COMP 2024)* is part of the SMT Workshop 2024, affiliated with CAV-36. The SMT Workshop will include a block of time to present the competitors and results of the competition.

Key dates

- **April 13** Deadline for new benchmark contributions
- **May 11** Final versions of competition tools (used by the organizers to run the participating solvers)
- **May 27** Deadline for first versions of participating solvers (for all tracks), including preliminary system descriptions
- **June 8** Deadline for final versions of participating solvers, including final system descriptions
- **July 22–23** SMT Workshop (presentation of results)

Organizers

- Martin Bromberger (chair) - MPI für Informatik, Germany
- François Bobot - CEA List, France
- Martin Jonáš - Masaryk University, Czechia

SMT-COMP 2024 is organized under the direction of the SMT Steering Committee.

Acknowledgment

Understanding SMT Solvers

Real-World Applications

- Software verification and testing
- Hardware design verification, equivalence checking
- Automated theorem proving
- Constraint Satisfaction Problems (CSPs): Scheduling, Resource Allocation
- Program Synthesis, Automated Programming
- ....
- *Particle physics?*

First order logic (FOL)

- Expressive formal language system that breaks statements down into
  - Things
    - Constants: x, y
    - Functions: $f(x), g(x)$
  - Relationships
    - Predicates: $\texttt{assert}(x > y)$
  - Connectives
    - &&, ||, !
  - Quantifiers
    - $\exists, \forall$

Example of FOL:

- Every electron has a negative charge $\Rightarrow \forall x, (Electron(x) \rightarrow Charge(x, -1))$

Non-Example of FOL:

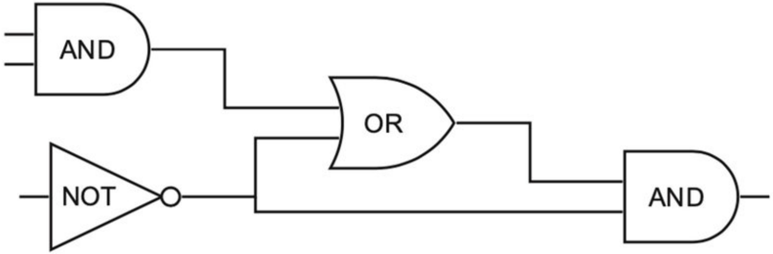
- There is a set of particles that behaves according to the Pauli exclusion principle $\Rightarrow \exists S, (Set(S) \wedge \forall x, \forall y, (x \in S \wedge y \in S \wedge x \neq y \rightarrow \neg SameState(x, y)))$

Notation

- Conjunction - &&, AND, $\wedge$, $*$
- Disjunction - $||$, OR, $\vee$, $+$
- Negation - !, NOT, $\neg$
- Conjunctive normal form (CNF) - conjunction of one or more disjunctions of one or more literals
    - Example $(A \vee B \vee C) \wedge (\neg D \vee E \neg F)$

Satisfiability (SAT) problems

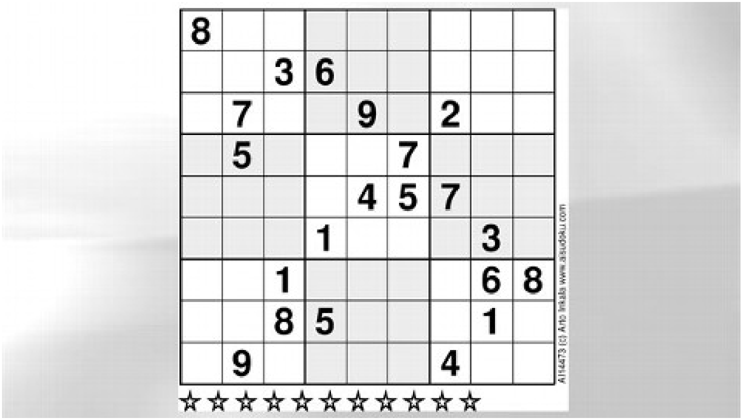
- Can we satisfy all the constraints?
  - $\exists a, b, c, d \in \mathbb{B} : (a \vee b \vee \neg c) \wedge (\neg a \vee d)$
- Is the circuit going to be satisfied?
  -



Modulo Theories

`bool` is not enough:

- $\exists x, y : 0.437x + \sqrt{y} = 23.8$
- $\exists x, y : x^2 > 3.7y + 85$

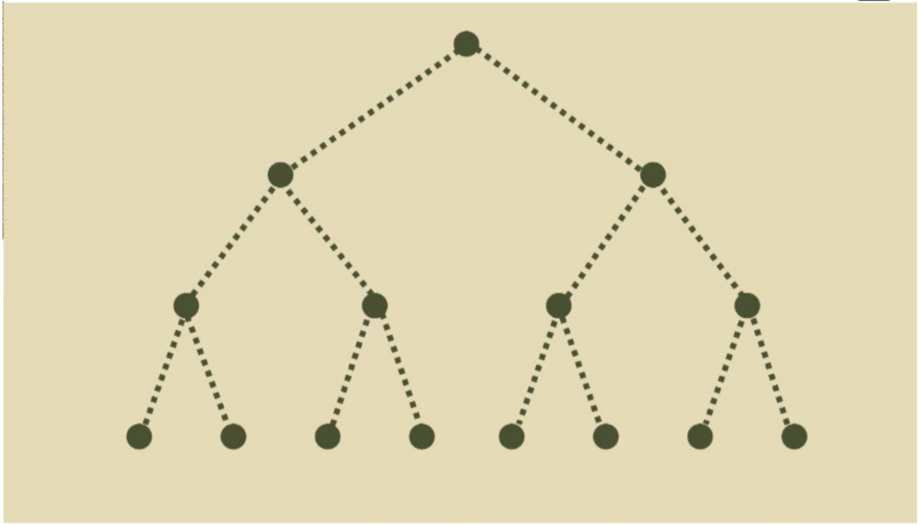


8								
		3	6					
	7			9		2		
	5				7			
				4	5	7		
			1				3	
		1					6	8
		8	5				1	
	9					4		

☆☆☆☆☆☆☆☆☆☆☆

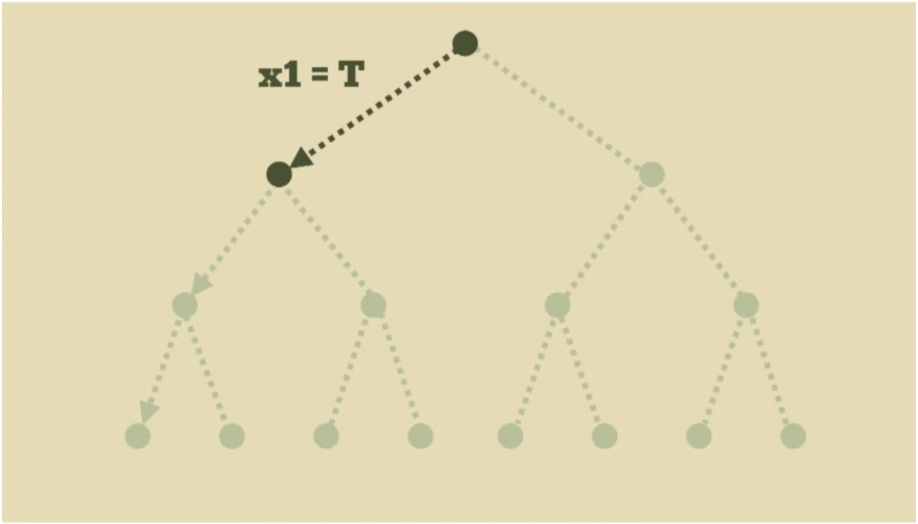
AI14473 (c) Arto Inkala www.aisudoku.com

Basic SAT solving: DPLL (Davis–Putnam–Logemann–Loveland)

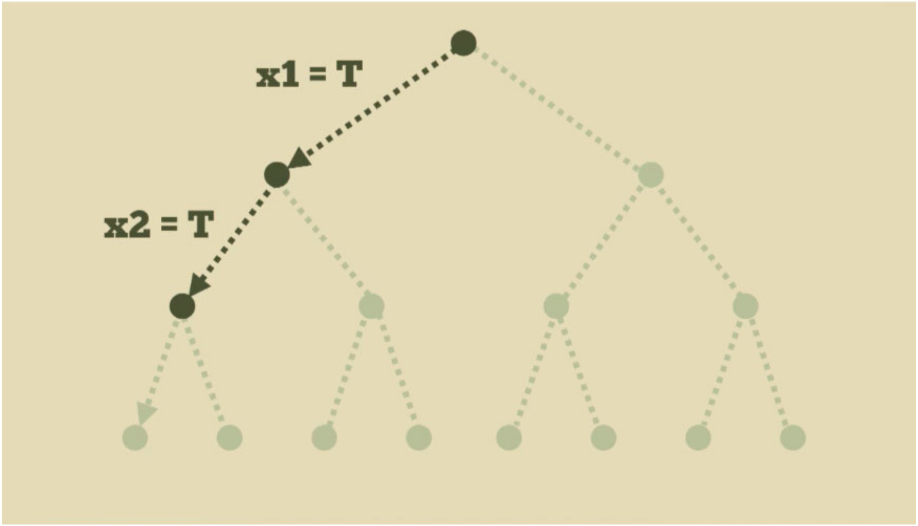


Taken from Jon Smock, 2016

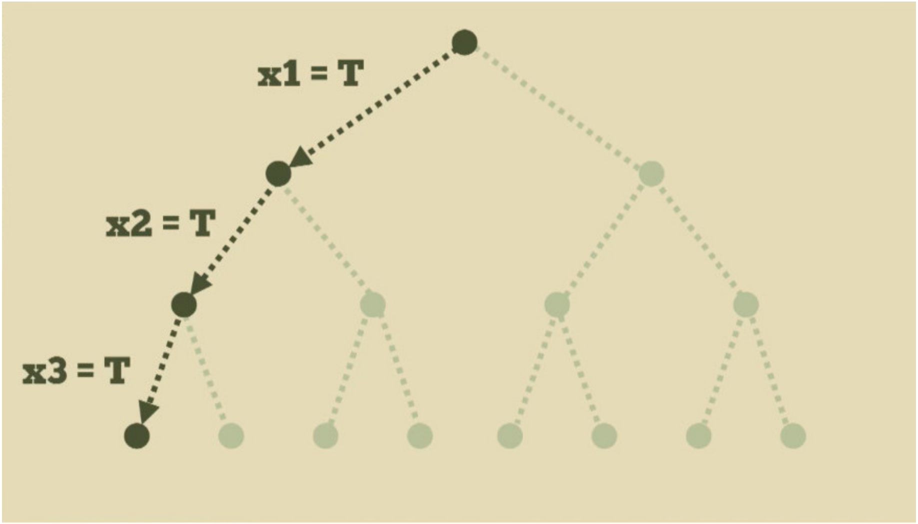
Basic SAT solving: DPLL (Davis–Putnam–Logemann–Loveland)



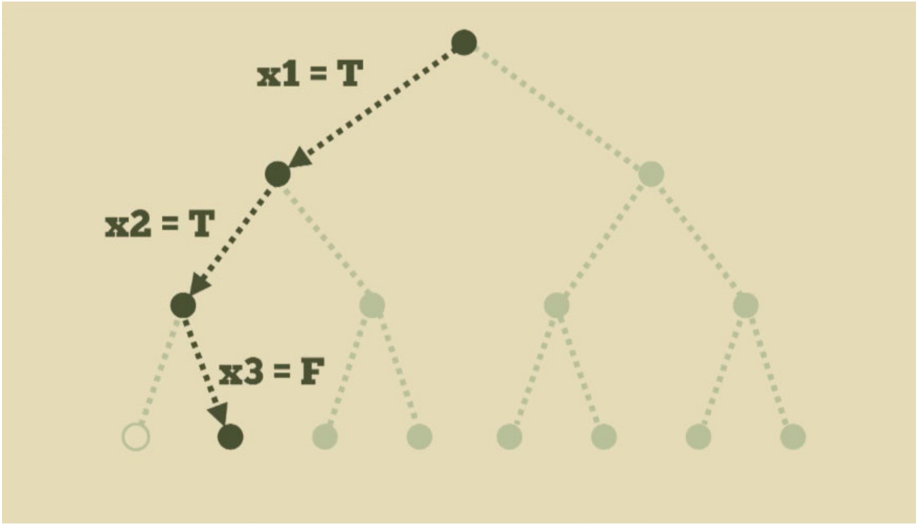
Basic SAT solving: DPLL (Davis–Putnam–Logemann–Loveland)



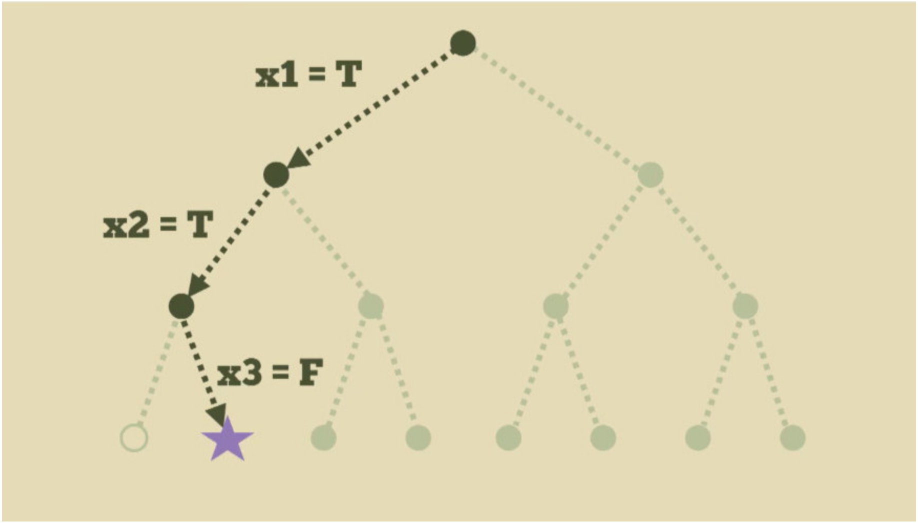
Basic SAT solving: DPLL (Davis–Putnam–Logemann–Loveland)



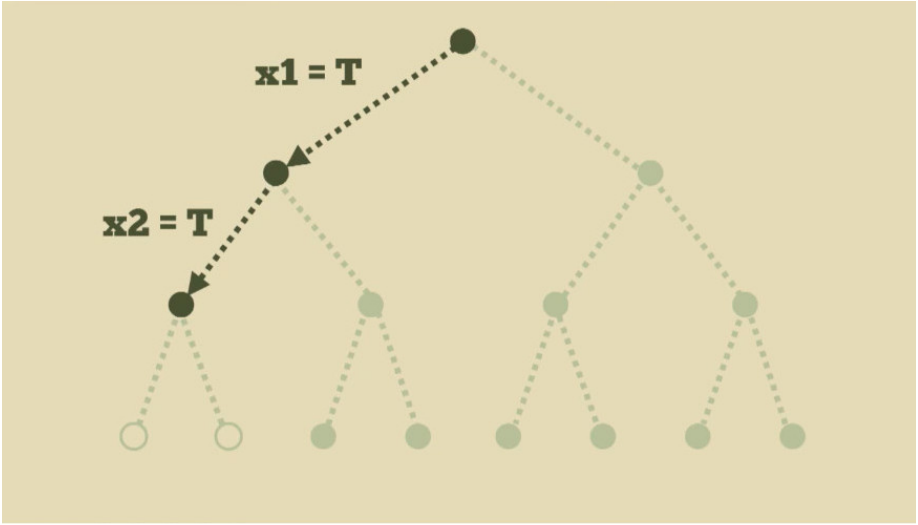
Basic SAT solving: DPLL (Davis–Putnam–Logemann–Loveland)



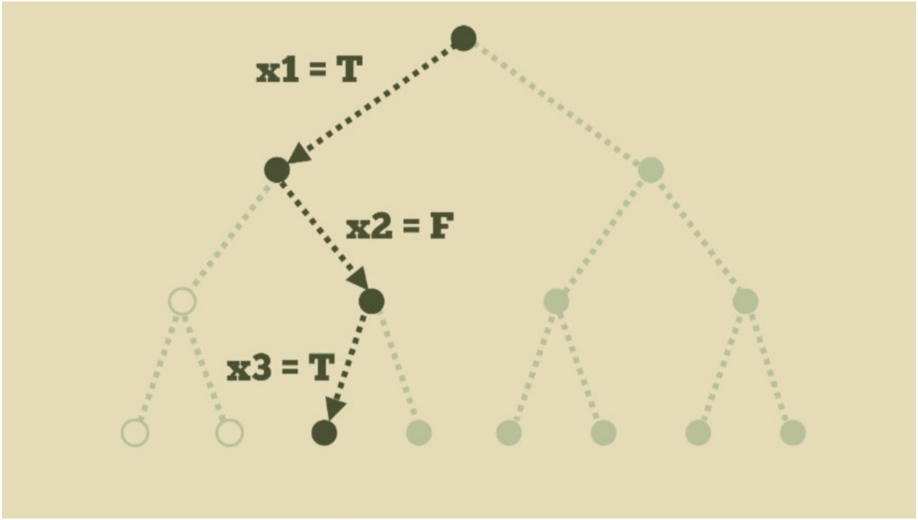
Basic SAT solving: DPLL (Davis–Putnam–Logemann–Loveland)



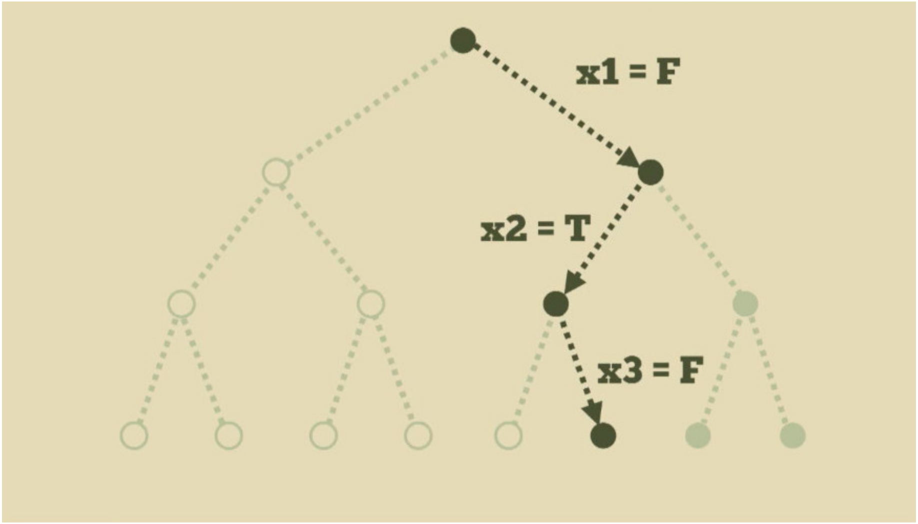
Basic SAT solving: DPLL (Davis–Putnam–Logemann–Loveland)



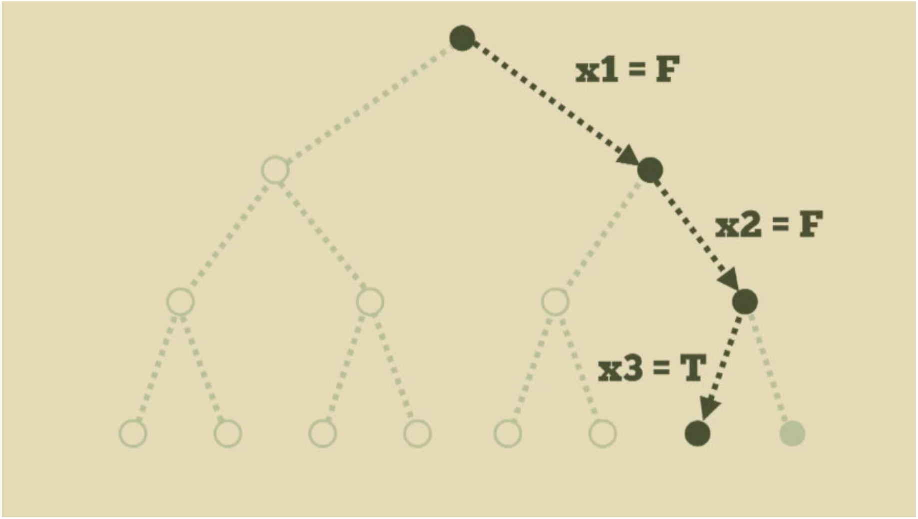
Basic SAT solving: DPLL (Davis–Putnam–Logemann–Loveland)



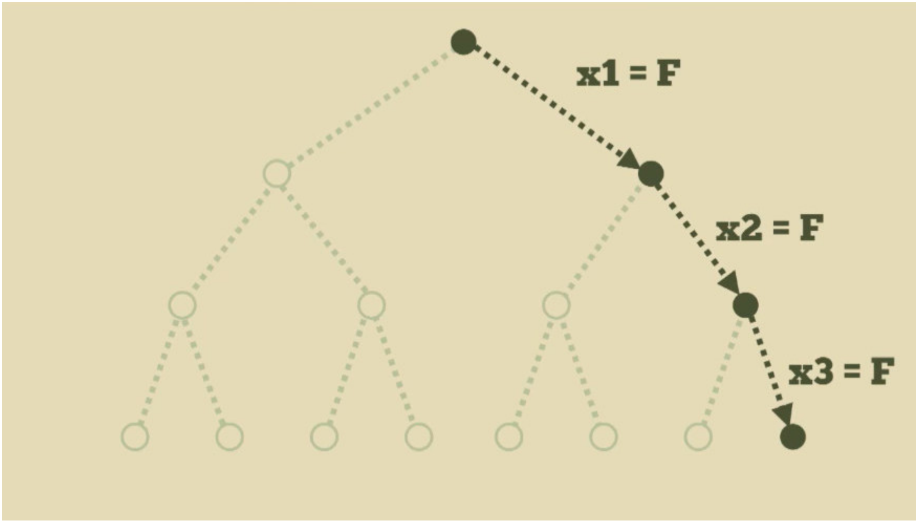
Basic SAT solving: DPLL (Davis–Putnam–Logemann–Loveland)



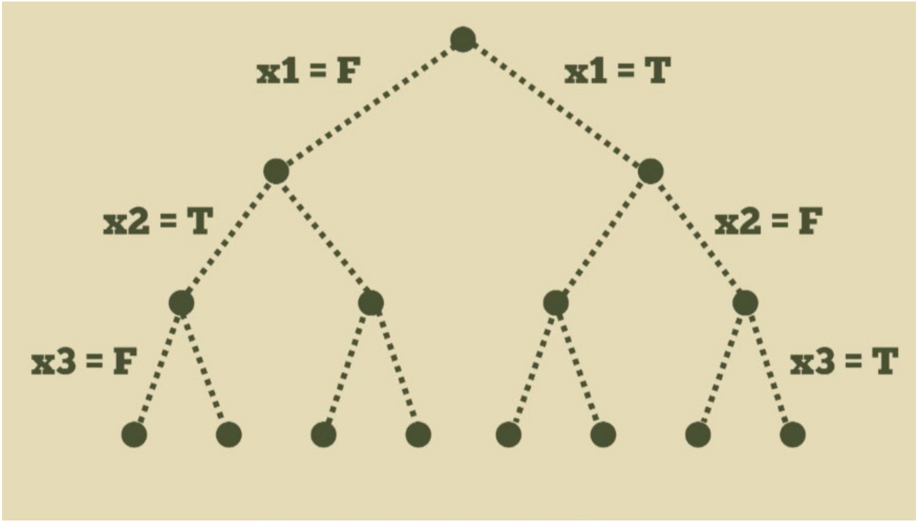
Basic SAT solving: DPLL (Davis–Putnam–Logemann–Loveland)



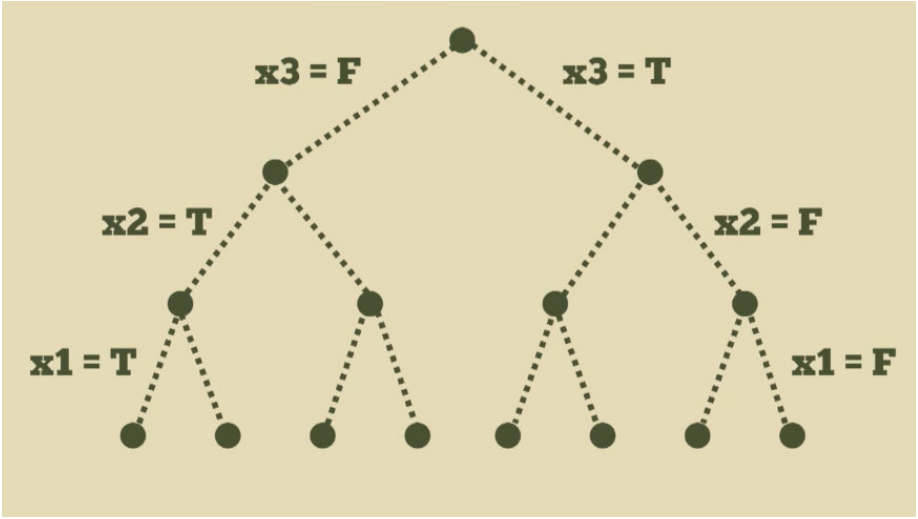
Basic SAT solving: DPLL (Davis–Putnam–Logemann–Loveland)



Basic SAT solving: DPLL (Davis–Putnam–Logemann–Loveland)



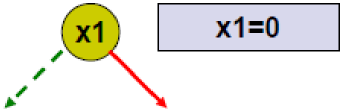
Basic SAT solving: DPLL (Davis–Putnam–Logemann–Loveland)



SAT solving: Conflict-driven clause learning (CDCL)

Step 1

x1 + x4
x1 + x3’ + x8’
x1 + x8 + x12
x2 + x11
x7’ + x3’ + x9
x7’ + x8 + x9’
x7 + x8 + x10’
x7 + x10 + x12’

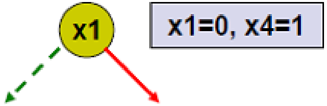


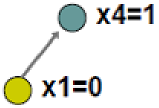
● x1=0

SAT solving: Conflict-driven clause learning (CDCL)

Step 2

x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'

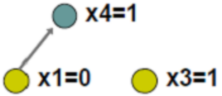


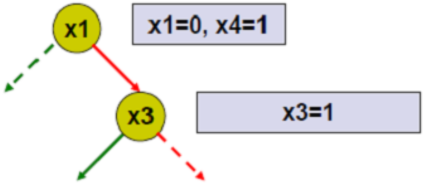


SAT solving: Conflict-driven clause learning (CDCL)

Step 3

x1 + x4
x1 + x3’ + x8’
x1 + x8 + x12
x2 + x11
x7’ + x3’ + x9
x7’ + x8 + x9’
x7 + x8 + x10’
x7 + x10 + x12’

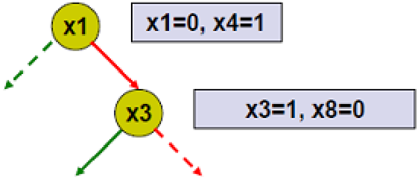


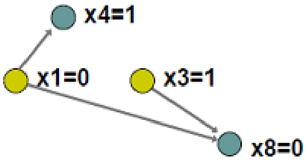


SAT solving: Conflict-driven clause learning (CDCL)

Step 4

x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'

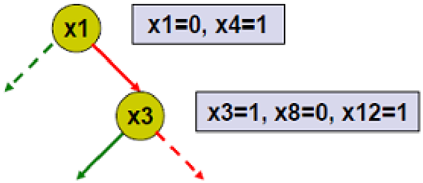


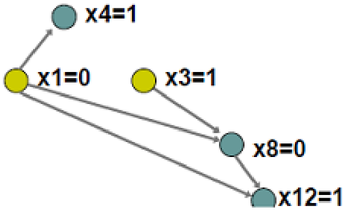


SAT solving: Conflict-driven clause learning (CDCL)

Step 5

x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'

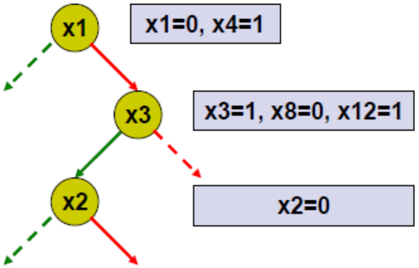


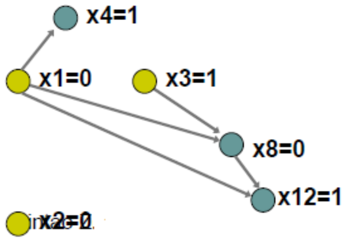


SAT solving: Conflict-driven clause learning (CDCL)

Step 6

x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'

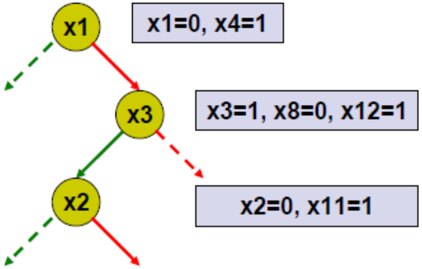


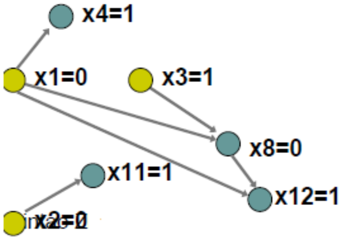


SAT solving: Conflict-driven clause learning (CDCL)

Step 7

x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'

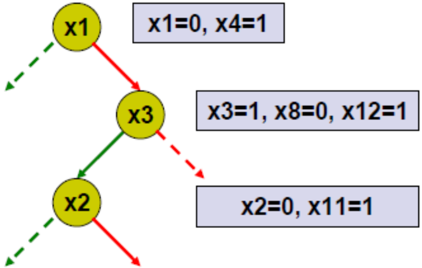


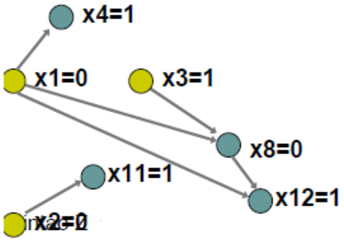


SAT solving: Conflict-driven clause learning (CDCL)

Step 8

x1 + x4
x1 + x3’ + x8’
x1 + x8 + x12
x2 + x11
x7’ + x3’ + x9
x7’ + x8 + x9’
x7 + x8 + x10’
x7 + x10 + x12’

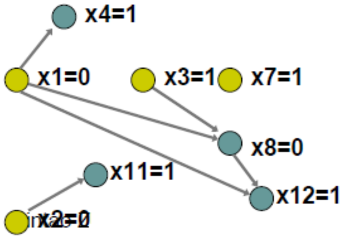


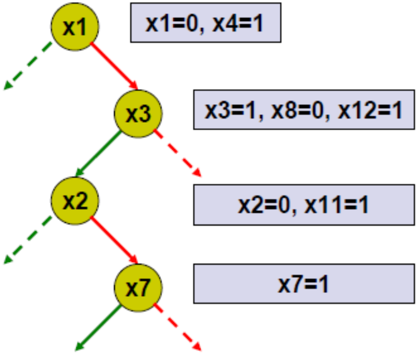


SAT solving: Conflict-driven clause learning (CDCL)

Step 9

x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'

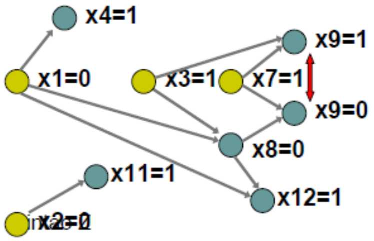


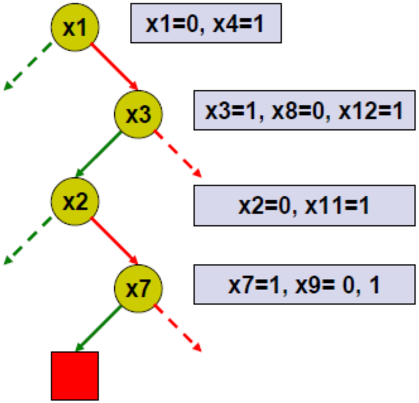


SAT solving: Conflict-driven clause learning (CDCL)

Step 10

x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'

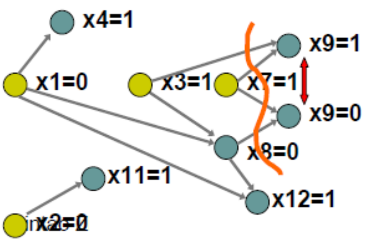


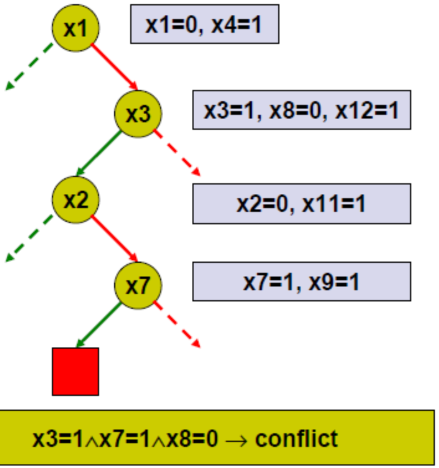


SAT solving: Conflict-driven clause learning (CDCL)

x1 + x4
x1 + x3’ + x8’
x1 + x8 + x12
x2 + x11
x7’ + x3’ + x9
x7’ + x8 + x9’
x7 + x8 + x10’
x7 + x10 + x12’

Step 11





SAT solving: Conflict-driven clause learning (CDCL)

If a implies b, then b' implies a'

Step 12

$x3=1 \wedge x7=1 \wedge x8=0 \rightarrow$ conflict

Not conflict $\rightarrow (x3=1 \wedge x7=1 \wedge x8=0)'$

true $\rightarrow (x3=1 \wedge x7=1 \wedge x8=0)'$

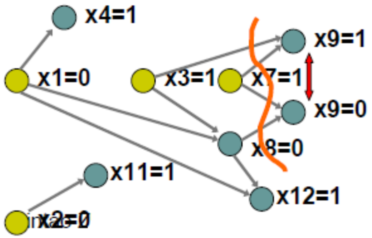
$(x3=1 \wedge x7=1 \wedge x8=0)'$

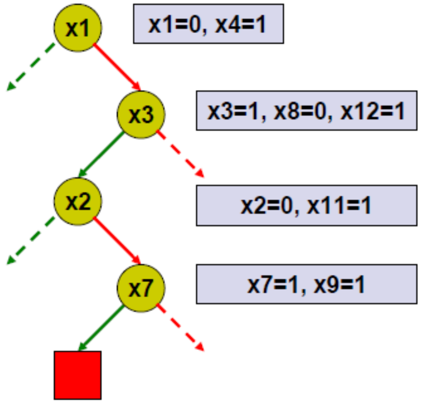
$\boxed{(x3' + x7' + x8)}$

SAT solving: Conflict-driven clause learning (CDCL)

x1 + x4
x1 + x3’ + x8’
x1 + x8 + x12
x2 + x11
x7’ + x3’ + x9
x7’ + x8 + x9’
x7 + x8 + x10’
x7 + x10 + x12’

Step 13



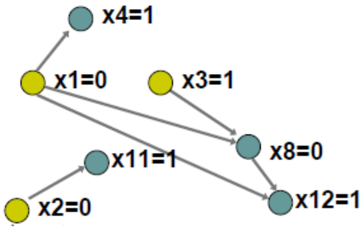


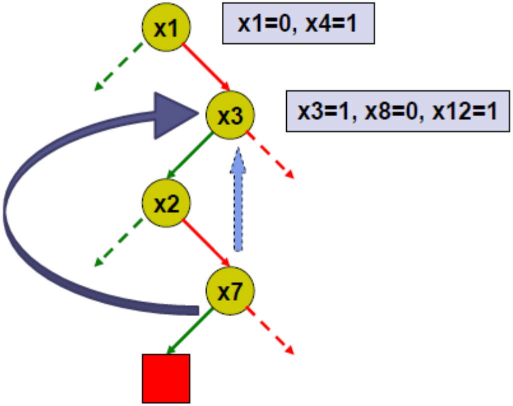
$x3=1 \wedge x7=1 \wedge x8=0 \rightarrow$ conflict
Add conflict clause: x3’+x7’+x8

SAT solving: Conflict-driven clause learning (CDCL)

x1 + x4
x1 + x3’ + x8’
x1 + x8 + x12
x2 + x11
x7’ + x3’ + x9
x7’ + x8 + x9’
x7 + x8 + x10’
x7 + x10 + x12’
$\boxed{\text{x3’ + x8 + x7’}}$

Step 14



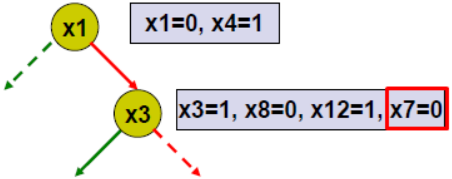


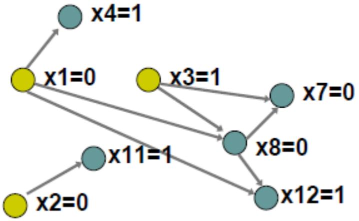
Backtrack to the decision level of x3=1:
x7 = 0

SAT solving: Conflict-driven clause learning (CDCL)

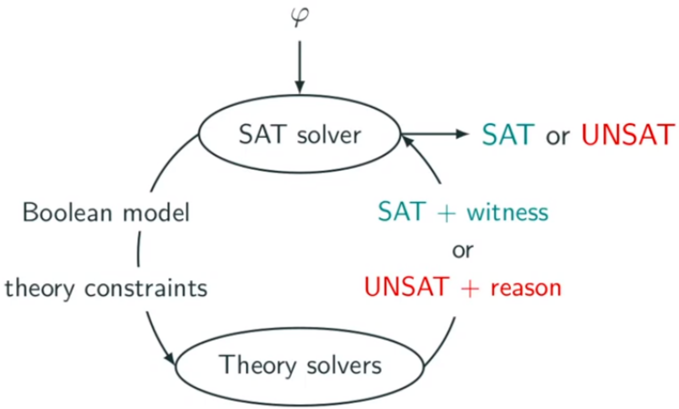
x1 + x4
x1 + x3’ + x8’
x1 + x8 + x12
x2 + x11
x7’ + x3’ + x9
x7’ + x8 + x9’
x7 + x8 + x10’
x7 + x10 + x12’
$\boxed{\text{x3’ + x8 + x7’}}$

Step 15



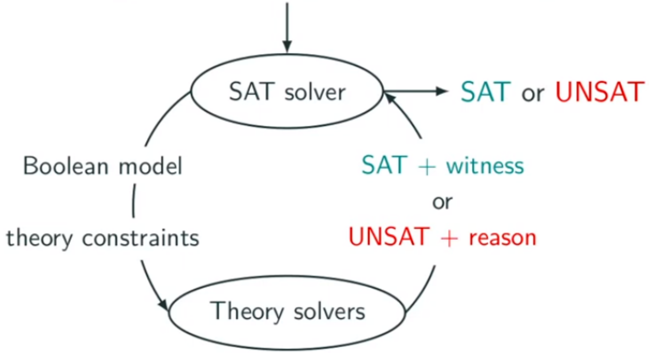


Basic princple of a SMT solver

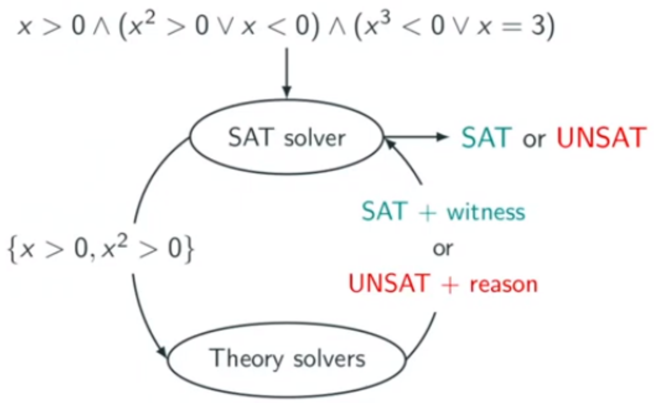


Basic princple of a SMT solver

$$x > 0 \land (x^2 > 0 \lor x < 0) \land (x^3 < 0 \lor x = 3)$$



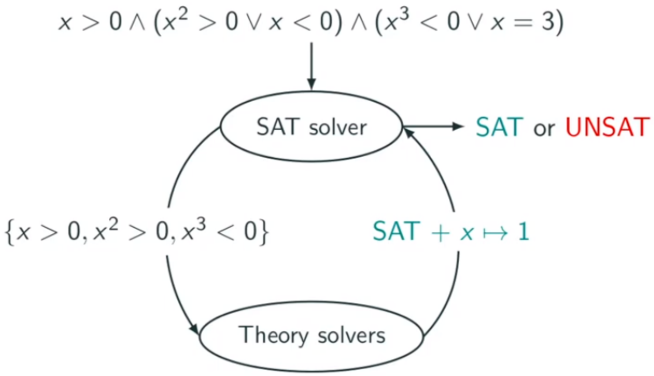
Basic princple of a SMT solver



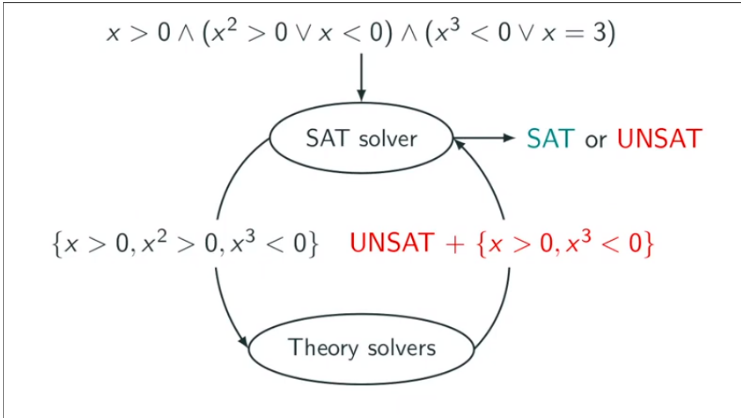
Basic princple of a SMT solver

$$\begin{tikzcd}
x > 0 \wedge (x^2 > 0 \vee x < 0) \wedge (x^3 < 0 \vee x = 3) \arrow[d] & \\
\text{SAT solver} \arrow[r] \arrow[d, bend right=60, "\{x > 0, x^2 > 0\}"'] & \text{SAT or UNSAT} \\
\text{Theory solvers} \arrow[u, bend right=60, "\text{SAT} + x \mapsto 1"'] &
\end{tikzcd}$$

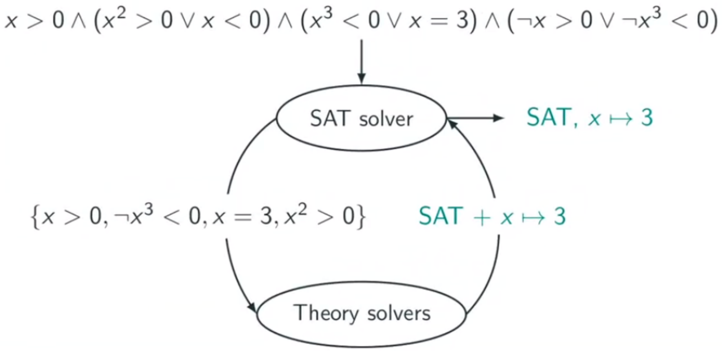
Basic princple of a SMT solver


$x > 0 \wedge (x^2 > 0 \vee x < 0) \wedge (x^3 < 0 \vee x = 3)$
SAT solver
SAT or UNSAT
$\{x > 0, x^2 > 0, x^3 < 0\}$
SAT + $x \mapsto 1$
Theory solvers

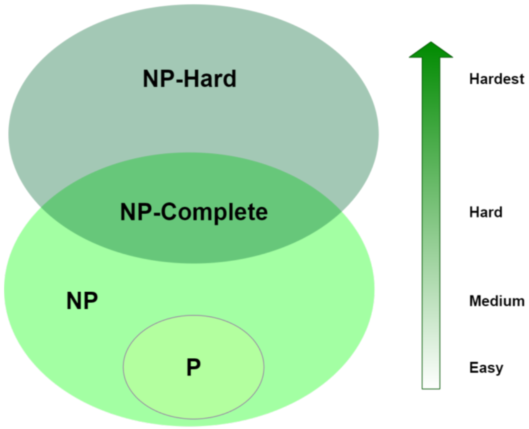
Basic princple of a SMT solver


$x > 0 \wedge (x^2 > 0 \vee x < 0) \wedge (x^3 < 0 \vee x = 3)$
SAT solver
SAT or UNSAT
$\{x > 0, x^2 > 0, x^3 < 0\}$
UNSAT + $\{x > 0, x^3 < 0\}$
Theory solvers

Basic princple of a SMT solver

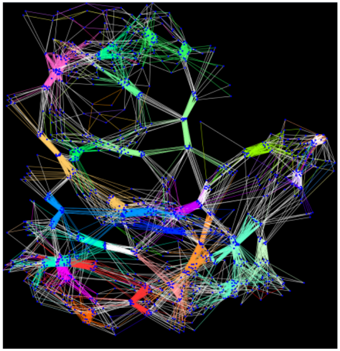

$x > 0 \wedge (x^2 > 0 \vee x < 0) \wedge (x^3 < 0 \vee x = 3) \wedge (\neg x > 0 \vee \neg x^3 < 0)$
SAT solver
SAT, $x \mapsto 3$
$\{x > 0, \neg x^3 < 0, x = 3, x^2 > 0\}$
SAT + $x \mapsto 3$
Theory solvers

Algorithmic complexity

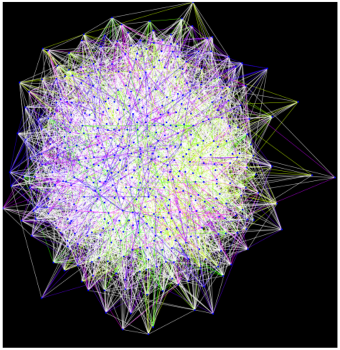


SAT is a NP-Complete problem, SMT is a NP-Hard problem...

Algorithmic complexity



(a) Industrial instance: *aes_16_10_keyfind_3*



(b) Random instance: *unif-k3-r4.267-v421-c1796-S4839562527790587617*

SATGraf: Evolution of SAT formula structure in solvers

- The most problems arising in the real life are solvable in a foreseeable time...

z3 - satisfiability modulo theories (SMT) solver

Demonstration of `z3-solver` in action:

```
from z3 import *
Vacation, Work = Bools('Vacation Work')
s = Solver()
s.add(Or(Vacation, Work),
Or(Not(Vacation), Work),
Or(Not(Vacation), Not(Work)))
print(s.check())
print(s.model())
```

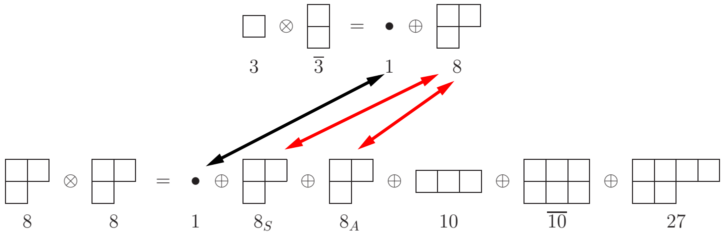
Output:

```
sat
[Vacation = False, Work = True]
```

z3 - Another example: 8 Queens

```
from z3 import *

# We know each queen must be in a different row.
# So, we represent each queen by a single integer: the column position
Q = [ Int('Q_%i' % (i + 1)) for i in range(8) ]

# Each queen is in a column {1, ... 8 }
val_c = [ And(1 <= Q[i], Q[i] <= 8) for i in range(8) ]

# At most one queen per column
col_c = [ Distinct(Q) ]

# Diagonal constraint
diag_c = [ If(i == j,
              True,
              And(Q[i] - Q[j] != i - j, Q[i] - Q[j] != j - i))
           for i in range(8) for j in range(i) ]

solve(val_c + col_c + diag_c)
```

Multiplet decomposition and matching for $q\bar{q} \to gg$ in $\mathrm{SU}(3)$

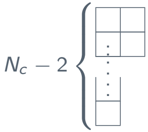


An orthogonal multiplet basis for $g(p_a) + g(p_b) \to \bar{q}(p_1) + q(p_2)$:

$$c_1 = \delta^{ab}\delta_{21}, \quad c_2 = d^{abc}T^c_{21}, \quad c_3 = if^{abc}T^c_{21}$$

- How to generate such a basis for an arbitrary $2 \to n$ process?

Automatic generation of color bases

- There are number of `Mathematica` packages for Young tableaux manipulation
  - `LieART` [Feger et al, 2019]
  - `xActs` [Nutma, 2013]
  - ...
- They only work for fixed value of N_c
  - Impossible to generate e.g. 

, which is 0 for $SU(3) \otimes SU(3)$
  - The performance scales very badly with increasing number of tableaux

Multiplet decomposition for 11 gluons

How can we generate something like this?

$$\begin{aligned}
8^{\otimes 11} = {} & 614000 \cdot 1 \oplus 3609760 \cdot 8 \oplus 3307040 \cdot \overline{10} \oplus 3307040 \cdot 10 \oplus 7247504 \cdot 27 \\
& \oplus 2596176 \cdot \overline{28} \oplus 2596176 \cdot 28 \oplus 6165720 \cdot \overline{35} \oplus 6165720 \cdot 35 \oplus 686070 \cdot \overline{55} \\
& \oplus 686070 \cdot 55 \oplus 8194065 \cdot 64 \oplus 2766060 \cdot \overline{80} \oplus 2766060 \cdot 80 \oplus 6049890 \cdot \overline{81} \\
& \oplus 6049890 \cdot 81 \oplus 61710 \cdot \overline{91} \oplus 61710 \cdot 91 \oplus 6030750 \cdot 125 \oplus 1320 \cdot \overline{136} \\
& \oplus 1320 \cdot 136 \oplus 433422 \cdot \overline{143} \oplus 433422 \cdot 143 \oplus 3811500 \cdot \overline{154} \oplus 3811500 \cdot 154 \\
& \oplus 1607364 \cdot \overline{162} \oplus 1607364 \cdot 162 \oplus 3035780 \cdot 216 \oplus 20460 \cdot \overline{224} \oplus 20460 \cdot 224 \\
& \oplus 1622720 \cdot \overline{260} \oplus 1622720 \cdot 260 \oplus 142780 \cdot \overline{270} \oplus 142780 \cdot 270 \oplus 592900 \cdot \overline{280} \\
& \oplus 592900 \cdot 280 \oplus 132 \cdot \overline{323} \oplus 132 \cdot 323 \oplus 1051666 \cdot 343 \oplus 466290 \cdot \overline{405}_{(13,8)} \\
& \oplus 466290 \cdot 405_{(13,5)} \oplus 2970 \cdot \overline{405}_{(16,14)} \oplus 2970 \cdot 405_{(16,2)} \oplus 140360 \cdot \overline{440} \\
& \oplus 140360 \cdot 440 \oplus 27148 \cdot \overline{442} \oplus 27148 \cdot 442 \oplus 244574 \cdot 512 \oplus 87010 \cdot \overline{595} \\
& \oplus 87010 \cdot 595 \oplus 165 \cdot \overline{640} \oplus 165 \cdot 640 \oplus 20196 \cdot \overline{648} \oplus 20196 \cdot 648 \oplus 2750 \cdot \overline{665} \\
& \oplus 2750 \cdot 665 \oplus 35970 \cdot 729 \oplus 9680 \cdot \overline{836} \oplus 9680 \cdot 836 \oplus 1540 \cdot \overline{910} \oplus 1540 \cdot 910 \\
& \oplus 110 \cdot \overline{945} \oplus 110 \cdot 945 \oplus 2980 \cdot 1000 \oplus 540 \cdot \overline{1134} \oplus 540 \cdot 1134 \oplus 44 \cdot \overline{1232} \\
& \oplus 44 \cdot 1232 \oplus 110 \cdot 1331 \oplus 10 \cdot \overline{1495} \oplus 10 \cdot 1495 \oplus 1 \cdot 1728 \quad = \quad 8\,589\,934\,592
\end{aligned}$$

Multiplet decomposition for 100 gluons

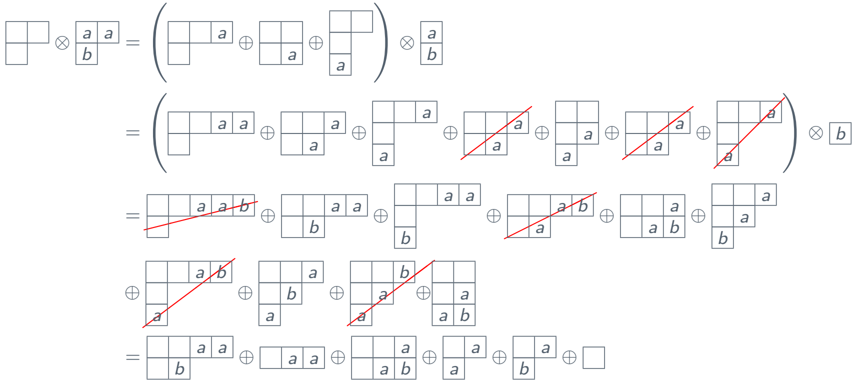
Or this?

$$\begin{aligned}
8^{\otimes 100} &= 334435009184561935119281727736623673012693743398748845163447438613371195381972168\,96 \otimes 1 \\
&\oplus 2572587806584635130066976463758529170267049506596346332697143938041693095231202432\,00 \otimes 8 \\
&\oplus 3092049948930685239427759764032876177490160094546038999708718630661556302925802640\,00 * \overline{10} \\
&\oplus 3092049948930685239427759764032876177490160094546038999708718630661556302925802640\,00 \otimes 10 \\
&\oplus 8133045977090587526241442014164163542386354479917916157325323744215623750775999588\,00 \otimes 27 \\
&\oplus 7400270348142806028788976869852519354326239069160475932818987762871878639568984200\,00 \otimes \overline{28} \\
&\oplus 7400270348142806028788976869852519354326239069160475932818987762871878639568984200\,00 \otimes 28 \\
&\oplus 10005538997495761783392858571640456666859204898869274947743791853823711417580338972\,00 \otimes \overline{35} \\
&\oplus 10005538997495761783392858571640456666859204898869274947743791853823711417580338972\,00 \otimes 35 \\
&\oplus 11485515663401859904530396675053136848792190723876158941078195014276331669906152000\,00 \otimes \overline{55} \\
&\oplus 11485515663401859904530396675053136848792190723876158941078195014276331669906152000\,00 \otimes 55 \\
&\oplus 17591872995964833737909650521160169089813317397209061842247507988335815611705170838\,40 \otimes 64 \\
&\oplus 18794839993080754465845632881249944381355088526899629530568881599851647125417936640\,00 \otimes \overline{80} \\
&\oplus \ldots \oplus \ldots \oplus 3759525 \otimes \overline{966735} \oplus 3759525 \otimes 966735 \oplus 15998400 \otimes \overline{969408} \oplus 15998400 \otimes 969408 \\
&\oplus 24497649 \otimes 970299 \oplus 156750 \otimes \overline{983060} \oplus 156750 \otimes 983060 \oplus 489951 \otimes \overline{984851} \oplus 489951 \otimes 984851 \\
&\oplus 4850 \otimes \overline{999100} \oplus 4850 \otimes 999100 \oplus 9900 \otimes 1000000 \oplus 99 \otimes \overline{1014849} \oplus 99 \otimes 1014849 \oplus 1 \otimes 1030301 \\
&= 2037035976334486086268445688409378161051468393665936250636140449354381299763336706183397376 \\
&= 2.04 \cdot 10^{90}
\end{aligned}$$

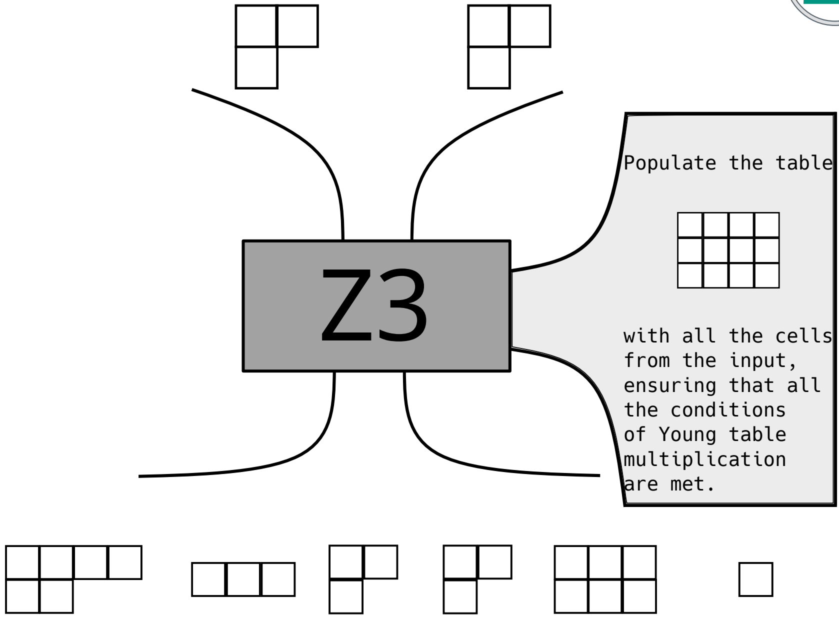
Multiplet decomposition for 100 gluons

- 873404864895945530640033598677182796408721441141854550663349444034725795666508301976296 tableaux in total
- Obtained in ~ 8 hours on a regular laptop
- The memory usage never exceeded 2 GB
- What's a secret?

Young tableaux multiplication: a textbook approach



Young tableaux multiplication: SMT solver approach



Populate the table

with all the cells from the input, ensuring that all the conditions of Young table multiplication are met.

Orthobase

`Orthobase` - A Python library to decompose the QCD processes in an orthogonal color basis:

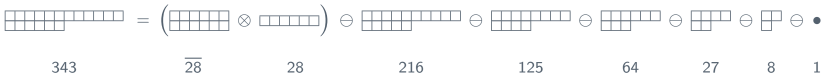
- Construct multiplet projectors for the decomposition of adjoint representation products in SU(N_c) groups
- Efficient algorithms implemented using the Z3 solver and FORM symbolic manipulation
- Comprehensive set of methods for working with Young tableaux
- Easy-to-use Python interface
- Described in [arXiv:2404.02443]
- Install using:
  `pip install --user OrthoBase`
- The manual is emerging at: `https://orthobase.readthedocs.io`

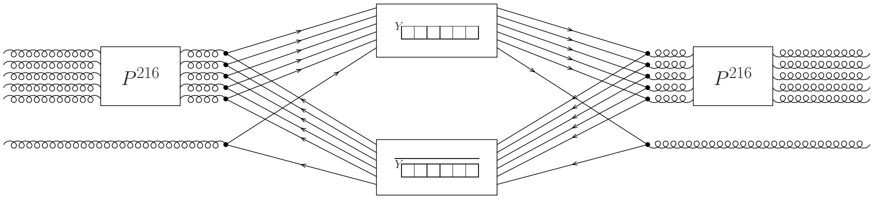
Orthobase - Usage example

```
#!/usr/bin/env python3
from OrthoBase import YoungTools as YT

#Define the number of colors
Nc = 3
#Define the SU(3) octet (gluon)
g = YT.YoungTableau([Nc-1,1],Nc)

#Perform the decomposition of 25 gluons
multiplets = g**25
#Print information about the resulting multiplets
multiplets.print():

#Get the list of all 162-plets
y_10_list = multiplets["162"]

#Obtain the conjugate of the first one
y_10_1_conj = y_10_list.conjugate()

#decompose as a quark antiquark pair
y_10_1_conj.decompose()
print(y_10_1_conj.decomposition)
```

Example: Construction of P^{343} projector

- We follow the strategy outlined in [Keppeler, Sjödahl, 2012]





$$
\begin{aligned}
P^{343}_{\mu_1\mu_2\mu_3\mu_4\mu_5\mu_6,\nu_1,\nu_2,\nu_3,\nu_4,\nu_5,\nu_6} =& P^{216}_{\mu_1\mu_2\mu_3\mu_4\mu_5} T^{\mu_1}_{a_1 m_1} T^{\mu_2}_{a_2 m_2} T^{\mu_3}_{a_3 m_3} T^{\mu_4}_{a_4 m_4} T^{\mu_5}_{a_5 m_5} T^{\mu_6}_{a_6 m_6} \\
&\times P^{216}_{\nu_1\nu_2\nu_3\nu_4\nu_5} T^{\mu_1}_{b_1 n_1} T^{\mu_2}_{b_2 n_2} T^{\mu_3}_{b_3 n_3} T^{\mu_4}_{b_4 n_4} T^{\mu_5}_{b_5 n_5} T^{\mu_6}_{b_6 n_6} \\
&\times (\delta^{a_1 b_1}\delta^{a_2 b_2}\delta^{a_3 b_3}\delta^{a_4 b_4}\delta^{a_5 b_5}\delta^{a_6 b_6} + \text{all symm. permutations}) \\
&\times (\delta^{m_1 n_1}\delta^{m_2 n_2}\delta^{m_3 n_3}\delta^{m_4 n_4}\delta^{m_5 n_5}\delta^{m_6 n_6} + \text{all symm. permutations}) \\
&- \tilde{P}^{216} - \tilde{P}^{125} - \tilde{P}^{65} - \tilde{P}^{27} - \tilde{P}^{8} - \tilde{P}^{1}
\end{aligned}
$$

Another example: Phase-space generation

- Generate 2->6 phase space::

```
sol.add(dot(p3,p3)==1)
sol.add(dot(p4,p4)==16)
sol.add(dot(p5,p5)==16)
sol.add(dot(p6,p6)==16)
sol.add(dot(p7,p7)==16)
sol.add(dot(p8,p8)==16)
sol.add(p1[0]==4000)
sol.add(p1[1]==0)
sol.add(p1[2]==0)
sol.add(p1[3]==4000)
sol.add(p2[0]==4000)
sol.add(p2[1]==0)
sol.add(p2[2]==0)
sol.add(p2[3]==-4000)
for row in range(4):
  sol.add(p3[row]+p4[row]+p5[row]+p6[row]+p7[row]+p8[row]-p1[row]-p2[row]==0)
```

Summary

- SMT solvers offer powerful techniques for solving complex logical problems
- Applications in particle physics:
  - Automatic generation of orthogonal multiplet bases in SU(N_c)
  - Young tableaux manipulations
  - Construction of multiplet projectors
  - Phase-space generation
  - Simplified Spinor-Helicity formalism
  - ...
- Orthobase - A Python library for QCD color decompositionusing a novel approach:
  - Efficient algorithms using Z3 solver and FORM
  - Handles large-scale problems (e.g., 100 gluon multiplet decomposition, no hard limit)
- SMT solvers can be very helpful to optimize dealing with the complex logics arising in certain particle physics problems

Thanks for your attention!

Karlsruhe Institute of Technology