CIS 1921

Lecture 10: From CP to SAT

Ishaan Lal ilal@seas.upenn.edu

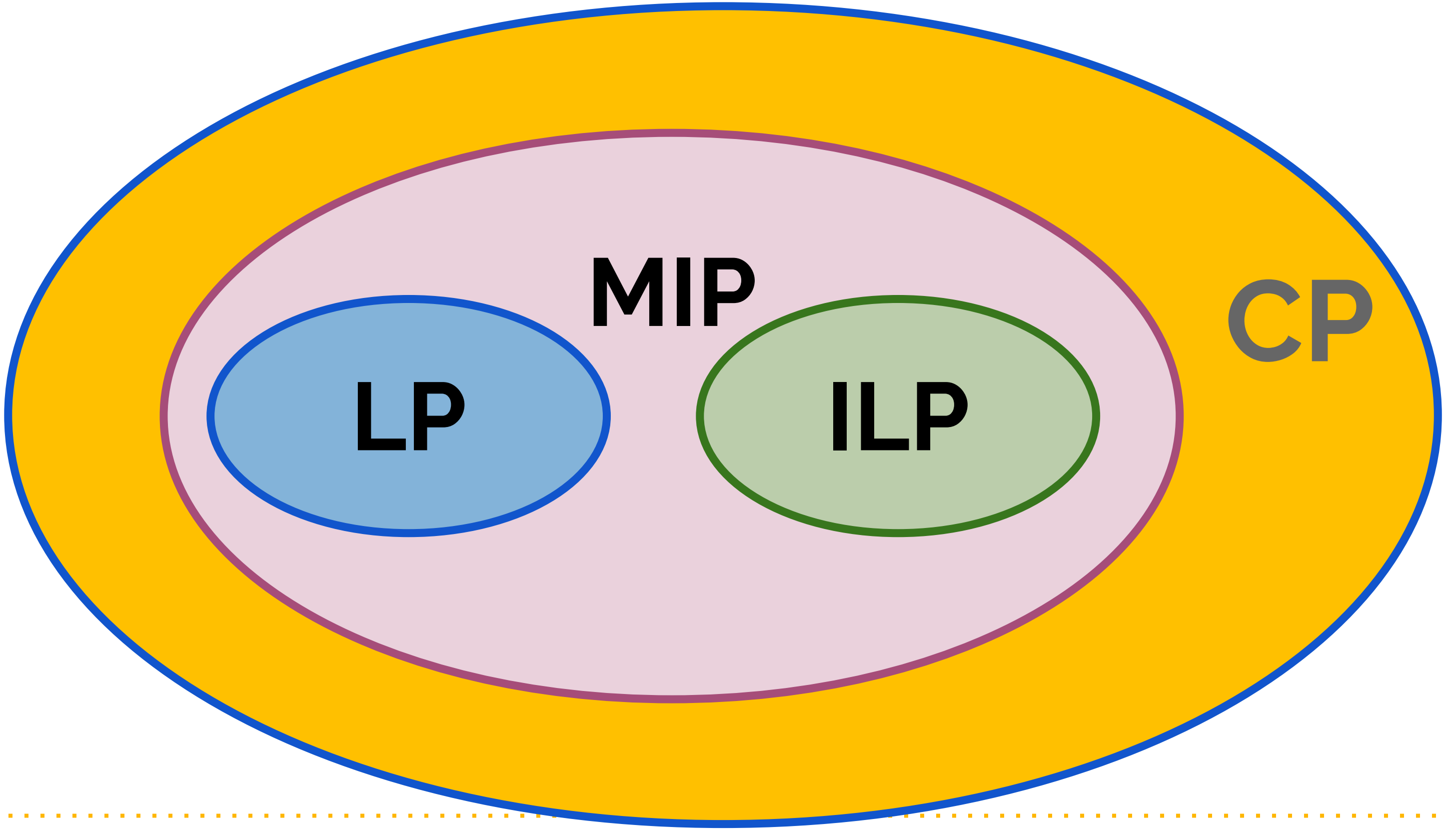
Logisitcs

- HW4 update. . .
- Project:
  - Feedback on proposal released on Gradescope
  - Checkpoint due 11/21 (~75% complete)
  - Please update us on any changes in vision to your project!

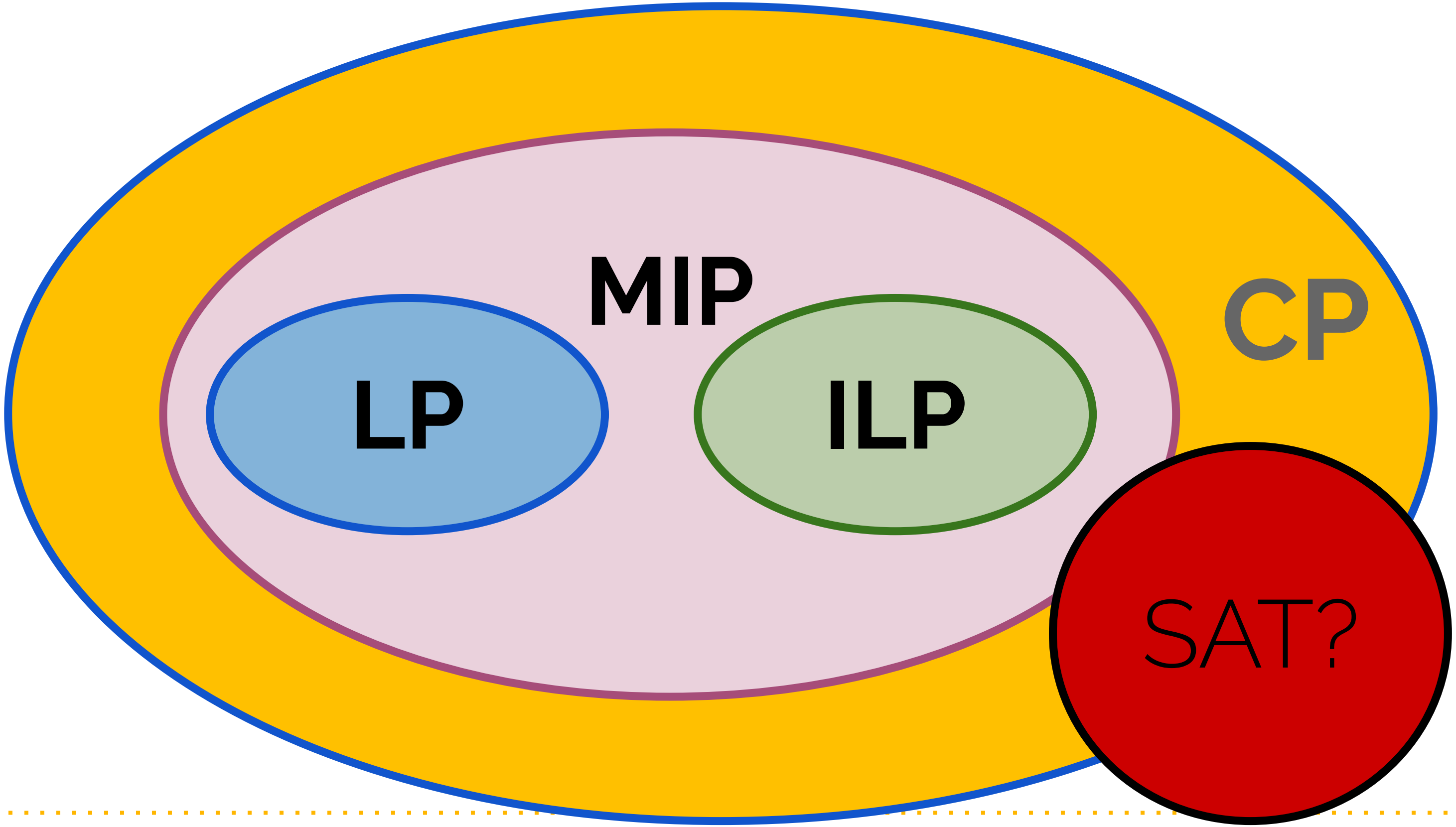
Today...

- **Back to SAT!**
- How does CP relate to SAT?

Where does SAT fit into the picture?

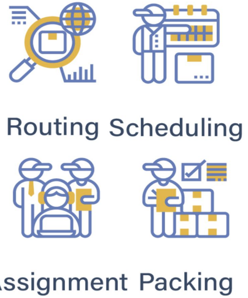


Where does SAT fit into the picture?



A loose analogy I've used

Constraint Program

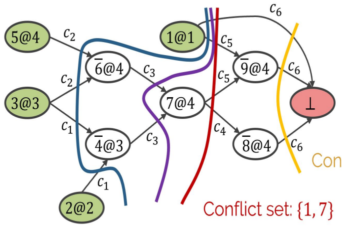


CP-SAT

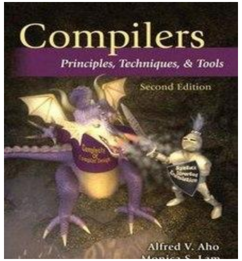
SAT Formula

p	q	p∧q
T	T	T
T	F	F
F	T	F
F	F	F

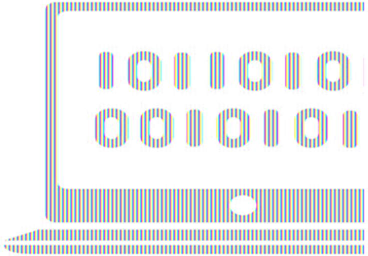
SAT Solver



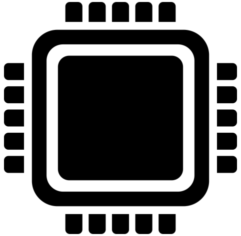
Programming Language



Compiler



Assembly Language



Hardware

Today...

- How does CP-SAT "compile" constraint programs into CNF-SAT formulas?
- Actually, that analogy is **wrong**!
  - CP-SAT does not just turn constraints into clauses and hand it off to a SAT solver...
  - We'll see it's more like a "conversation" btwn CP-SAT & solver
- Disclaimer: this is active research
  - Many details are necessarily left out, and any errors are mine
  - Thanks to: P. Stuckey, O. Ohrimenko, M. Codish, T. Feydy

Recap: CDCL

Conflict Driven Clause Learning

Who remembers?

Recap: CDCL

```
cdcl(φ):
    if unit_propagate() = CONFLICT: return UNSAT
    while not all variables have been set:
        let x = pick_variable()
        create new decision level; set x = T
        while unit_propagate() = CONFLICT:
            if level = 0: return UNSAT
            let (conflict_cls, assrt_lvl) = analyze_conflict()
            let φ = φ ∪ { conflict_cls }
            # discard all assignments after asserting level
            backjump(assrt_lvl)
    return SAT
```

Recap: CDCL

- Recall: CDCL = **Conflict Driven Clause Learning**
- **Incrementality:** CDCL solvers allow new clauses to be added during the search
- **Conflict analysis**
  - Build implication graph
  - Find set of literals that caused the conflict
  - Learn a new conflict clause

Recap: CDCL

c_1: $\bar{2} \vee \bar{3} \vee \bar{4}$

c_2: $\bar{3} \vee \bar{5} \vee \bar{6}$

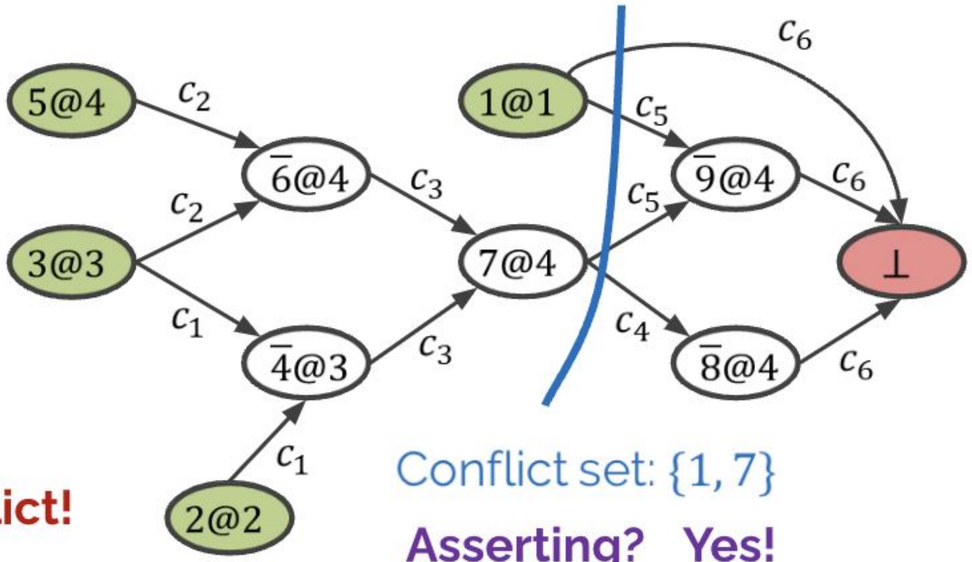
c_3: $4 \vee 6 \vee 7$

c_4: $\bar{7} \vee \bar{8}$

c_5: $\bar{1} \vee \bar{7} \vee \bar{9}$

c_6: $\bar{1} \vee 8 \vee 9$ **Conflict!**

c: $\bar{1} \vee \bar{7}$



Conflict set: {1, 7}

Asserting? Yes!

Level? 1

Background: CP Solvers

- We'll consider CP over discrete finite domains only (i.e., bounded integer vars)
- Need to understand a bit about how traditional finite domain solvers work first

IBM CPLEX

SWI Prolog

ECLiPSe

Background: CP Solvers

- Maintain a domain D that tracks the possible values for each variable
  - Doesn't need to be contiguous (e.g., $\{1,3,5\}$)
- Let $\min_D(x)$ and $\max_D(x)$ denote the min and max possible values for variable x in domain D
  - Initially $D(x) = [\mathrm{lb}(x)..\mathrm{ub}(x)]$ for each variable x

Bounds Consistency

- We say a constraint c involving variables $x_1, \ldots, x_n$ is **bounds consistent** with domain D if for each x_i:
  - it's possible to set $x_i = \min_D(x_i)$ and still satisfy c, and
  - it's possible to set $x_i = \max_D(x_i)$ and still satisfy c
- **Ex:** $D(x) = [4..7],\ D(y) = [1..5],\ D(z) = [-1..2]$ subject to $x = y + z$
  - $x = 4 \rightarrow y = 4,\ z = 0$ ✔
  - $x = 7 \rightarrow y = 5,\ z = 2$ ✔
  - $y = 1 \rightarrow x = 4,\ z = ?$ ❌ not bounds consistent!

In other words, "there exists a solution within the bounds when we set x_i = min and x_i = max"

Bounds Consistency

$$D(x) = [4..7], D(y) = [2..5], D(z) = [-1..2]$$

$$\text{subject to } x = y + z$$

$x = 4 \rightarrow y = 4,\ z = 0 \quad \checkmark$

$x = 7 \rightarrow y = 5,\ z = 2 \quad \checkmark$

$y = 2 \rightarrow x = 4,\ z = 2 \quad \checkmark$

$y = 5 \rightarrow x = 6,\ z = 1 \quad \checkmark$

$z = -1 \rightarrow x = 4,\ y = 5 \quad \checkmark$

$z = 2 \rightarrow x = 6,\ y = 4 \quad \checkmark$

Propagators

- A **propagator** for constraint c is an algorithm that accepts a domain D, and returns:
    - A new domain D' where c is bounds consistent with D'
    - Implications "explaining" the updated bounds in D'
- Different constraints have different propagation rules for finding D'

Propagator for $x = y + z$

- How to ensure bounds consistency for $x = y + z$?
- We can rewrite to isolate each variable:

$$x = y + z \qquad y = x - z \qquad z = x - y$$

- Now we can derive a pair of inequalities for each:
  - $x \geq \min_D(y) + \min_D(z)$ and $x \leq \max_D(y) + \max_D(z)$
  - $y \geq \min_D(x) - \max_D(z)$ and $y \leq \max_D(x) - \min_D(z)$
  - $z \geq \min_D(x) - \max_D(y)$ and $z \leq \max_D(x) - \min_D(y)$

Propagator for $x = y + z$

- What are "explanations"?
- **Ex:** $D(x) = [4..7],\ D(y) = [1..5],\ D(z) = [-1..2]$
- Since $y \geq \min_D(x) - \max_D(z) = 4 - 2 = 2$, we update the domain of y to $D'(y) = [2..5]$
- The explanation for this update is the implication:

$$(x \geq 4) \land (z \leq 2) \Rightarrow y \geq 2$$

Bounds Consistency

$$D(x) = [4..7], D(y) = [2..5], D(z) = [-1..2]$$

$$\text{subject to } x = y + z$$

$x = 4 \to y = 4,\ z = 0 \quad \checkmark$

$x = 7 \to y = 5,\ z = 2 \quad \checkmark$

$y = 2 \to x = 4,\ z = 2 \quad \checkmark$

$y = 5 \to x = 6,\ z = 1 \quad \checkmark$

$z = -1 \to x = 4,\ y = 5 \quad \checkmark$

$z = 2 \to x = 6,\ y = 4 \quad \checkmark$

Finite Domain Propagation

Many traditional CP solvers use **finite domain propagation**:

- Start with the initial domain D_0 specified by the user
- Try adding a new constraint c (e.g. assigning a variable)
- Repeatedly run all constraint propagators on D until:
  - A var has no possible values: BACKTRACK, add $\neg c$!
  - Nothing changes: add another constraint and repeat
- Does this sound familiar?

Ex: Finite Domain Propagation

$$D_0(x_1) = D_0(x_2) = D_0(x_3) = D_0(x_4) = D_0(x_5) = [1..4]$$

s.t. $x_2 \leq x_5$, $\text{AllDifferent}([x_1, x_2, x_3, x_4])$, $x_1 + x_2 + x_3 + x_4 \leq 9$

Ex: Finite Domain Propagation

$$D_0(x_1) = D_0(x_2) = D_0(x_3) = D_0(x_4) = D_0(x_5) = [1..4]$$

$$\text{s.t. } x_2 \leq x_5,\ \text{AllDifferent}([x_1, x_2, x_3, x_4]),\ x_1 + x_2 + x_3 + x_4 \leq 9$$

	$x_1 = 1$
$D(x_1)$	$\{1\}$
$D(x_2)$	$[1..4]$
$D(x_3)$	$[1..4]$
$D(x_4)$	$[1..4]$
$D(x_5)$	$[1..4]$

Domain D_1

Ex: Finite Domain Propagation

$$D_0(x_1) = D_0(x_2) = D_0(x_3) = D_0(x_4) = D_0(x_5) = [1..4]$$

s.t. $x_2 \le x_5$, $\text{AllDifferent}([x_1, x_2, x_3, x_4])$, $x_1 + x_2 + x_3 + x_4 \le 9$

	$x_1 = 1$	AllDiff
$D(x_1)$	$\{1\}$	$\{1\}$
$D(x_2)$	$[1..4]$	$[2..4]$
$D(x_3)$	$[1..4]$	$[2..4]$
$D(x_4)$	$[1..4]$	$[2..4]$
$D(x_5)$	$[1..4]$	$[1..4]$

Domain D_1

Ex: Finite Domain Propagation

$$D_0(x_1) = D_0(x_2) = D_0(x_3) = D_0(x_4) = D_0(x_5) = [1..4]$$

$$\text{s.t. } x_2 \le x_5,\ \text{AllDifferent}([x_1, x_2, x_3, x_4]),\ x_1 + x_2 + x_3 + x_4 \le 9$$

	$x_1 = 1$	AllDiff	$x_2 \le x_5$
$D(x_1)$	$\{1\}$	$\{1\}$	$\{1\}$
$D(x_2)$	$[1..4]$	$[2..4]$	$[2..4]$
$D(x_3)$	$[1..4]$	$[2..4]$	$[2..4]$
$D(x_4)$	$[1..4]$	$[2..4]$	$[2..4]$
$D(x_5)$	$[1..4]$	$[1..4]$	$[2..4]$

Domain D_1

Ex: Finite Domain Propagation

$$D_0(x_1) = D_0(x_2) = D_0(x_3) = D_0(x_4) = D_0(x_5) = [1..4]$$

s.t. $x_2 \leq x_5$, $\text{AllDifferent}([x_1, x_2, x_3, x_4])$, $x_1 + x_2 + x_3 + x_4 \leq 9$

	$x_1 = 1$	AllDiff	$x_2 \leq x_5$	$x_5 \leq 2$
$D(x_1)$	$\{1\}$	$\{1\}$	$\{1\}$	$\{1\}$
$D(x_2)$	$[1..4]$	$[2..4]$	$[2..4]$	$[2..4]$
$D(x_3)$	$[1..4]$	$[2..4]$	$[2..4]$	$[2..4]$
$D(x_4)$	$[1..4]$	$[2..4]$	$[2..4]$	$[2..4]$
$D(x_5)$	$[1..4]$	$[1..4]$	$[2..4]$	$\{2\}$

Domain D_1 **Domain D_2**

Ex: Finite Domain Propagation

$$D_0(x_1) = D_0(x_2) = D_0(x_3) = D_0(x_4) = D_0(x_5) = [1..4]$$

s.t. $x_2 \leq x_5$, $\text{AllDifferent}([x_1, x_2, x_3, x_4])$, $x_1 + x_2 + x_3 + x_4 \leq 9$

	$x_1 = 1$	AllDiff	$x_2 \leq x_5$	$x_5 \leq 2$	$x_2 \leq x_5$
$D(x_1)$	$\{1\}$	$\{1\}$	$\{1\}$	$\{1\}$	$\{1\}$
$D(x_2)$	$[1..4]$	$[2..4]$	$[2..4]$	$[2..4]$	$\{2\}$
$D(x_3)$	$[1..4]$	$[2..4]$	$[2..4]$	$[2..4]$	$[2..4]$
$D(x_4)$	$[1..4]$	$[2..4]$	$[2..4]$	$[2..4]$	$[2..4]$
$D(x_5)$	$[1..4]$	$[1..4]$	$[2..4]$	$\{2\}$	$\{2\}$

Domain D_1 (columns $x_1 = 1$, AllDiff, $x_2 \leq x_5$) **Domain D_2** (columns $x_5 \leq 2$, $x_2 \leq x_5$)

Ex: Finite Domain Propagation

$$D_0(x_1) = D_0(x_2) = D_0(x_3) = D_0(x_4) = D_0(x_5) = [1..4]$$

s.t. $x_2 \leq x_5$, $\text{AllDifferent}([x_1, x_2, x_3, x_4])$, $x_1 + x_2 + x_3 + x_4 \leq 9$

	$x_1 = 1$	AllDiff	$x_2 \leq x_5$	$x_5 \leq 2$	$x_2 \leq x_5$	AllDiff
$D(x_1)$	$\{1\}$	$\{1\}$	$\{1\}$	$\{1\}$	$\{1\}$	$\{1\}$
$D(x_2)$	$[1..4]$	$[2..4]$	$[2..4]$	$[2..4]$	$\{2\}$	$\{2\}$
$D(x_3)$	$[1..4]$	$[2..4]$	$[2..4]$	$[2..4]$	$[2..4]$	$[3..4]$
$D(x_4)$	$[1..4]$	$[2..4]$	$[2..4]$	$[2..4]$	$[2..4]$	$[3..4]$
$D(x_5)$	$[1..4]$	$[1..4]$	$[2..4]$	$\{2\}$	$\{2\}$	$\{2\}$
	Domain D_1			Domain D_2		

Ex: Finite Domain Propagation

$$D_0(x_1) = D_0(x_2) = D_0(x_3) = D_0(x_4) = D_0(x_5) = [1..4]$$

s.t. $x_2 \leq x_5$, $\text{AllDifferent}([x_1, x_2, x_3, x_4])$, $x_1 + x_2 + x_3 + x_4 \leq 9$

	$x_1 = 1$	AllDiff	$x_2 \leq x_5$	$x_5 \leq 2$	$x_2 \leq x_5$	AllDiff	$\sum \leq 9$
$D(x_1)$	$\{1\}$	$\{1\}$	$\{1\}$	$\{1\}$	$\{1\}$	$\{1\}$	$\{1\}$
$D(x_2)$	$[1..4]$	$[2..4]$	$[2..4]$	$[2..4]$	$\{2\}$	$\{2\}$	$\{2\}$
$D(x_3)$	$[1..4]$	$[2..4]$	$[2..4]$	$[2..4]$	$[2..4]$	$[3..4]$	$\{3\}$
$D(x_4)$	$[1..4]$	$[2..4]$	$[2..4]$	$[2..4]$	$[2..4]$	$[3..4]$	$\{3\}$
$D(x_5)$	$[1..4]$	$[1..4]$	$[2..4]$	$\{2\}$	$\{2\}$	$\{2\}$	$\{2\}$

Domain D_1 Domain D_2

Ex: Finite Domain Propagation

$$D_0(x_1) = D_0(x_2) = D_0(x_3) = D_0(x_4) = D_0(x_5) = [1..4]$$

s.t. $x_2 \leq x_5$, AllDifferent$([x_1, x_2, x_3, x_4])$, $x_1 + x_2 + x_3 + x_4 \leq 9$

	$x_1 = 1$	AllDiff	$x_2 \leq x_5$	$x_5 \leq 2$	$x_2 \leq x_5$	AllDiff	$\sum \leq 9$	AllDiff
$D(x_1)$	$\{1\}$	$\{1\}$	$\{1\}$	$\{1\}$	$\{1\}$	$\{1\}$	$\{1\}$	$\{1\}$
$D(x_2)$	$[1..4]$	$[2..4]$	$[2..4]$	$[2..4]$	$\{2\}$	$\{2\}$	$\{2\}$	$\{2\}$
$D(x_3)$	$[1..4]$	$[2..4]$	$[2..4]$	$[2..4]$	$[2..4]$	$[3..4]$	$\{3\}$	$\emptyset$
$D(x_4)$	$[1..4]$	$[2..4]$	$[2..4]$	$[2..4]$	$[2..4]$	$[3..4]$	$\{3\}$	$\emptyset$
$D(x_5)$	$[1..4]$	$[1..4]$	$[2..4]$	$\{2\}$	$\{2\}$	$\{2\}$	$\{2\}$	$\{2\}$

Domain D_1 **Domain D_2**

Ex: Finite Domain Propagation

$$D_0(x_1) = D_0(x_2) = D_0(x_3) = D_0(x_4) = D_0(x_5) = [1..4]$$

s.t. $x_2 \leq x_5$, $\text{AllDifferent}([x_1, x_2, x_3, x_4])$, $x_1 + x_2 + x_3 + x_4 \leq 9$

	$x_1 = 1$	AllDiff	$x_2 \leq x_5$	$x_5 > 2$	etc...
$D(x_1)$	$\{1\}$	$\{1\}$	$\{1\}$	$\{1\}$	
$D(x_2)$	$[1..4]$	$[2..4]$	$[2..4]$	$[2..4]$	
$D(x_3)$	$[1..4]$	$[2..4]$	$[2..4]$	$[2..4]$	
$D(x_4)$	$[1..4]$	$[2..4]$	$[2..4]$	$[2..4]$	
$D(x_5)$	$[1..4]$	$[1..4]$	$[2..4]$	$[3..4]$	

Backtrack!

Domain D_1 — Domain D_2

FD Propagation is Like DPLL

- Adding a constraint is like making a decision
- Running constraint propagators is like unit propagation
- Backtracking is like… backtracking
- So why don’t we try to just do this all in SAT?

Representing CP as SAT?

- First question: what are the boolean variables?
- **Attempt 1:** for each CP var x, create boolean variables $[\![x = i]\!]$ for $\mathrm{lb}(x) \leq i \leq \mathrm{ub}(x)$
  - Number of variables is linear in size of domain
  - Issue: need very long clauses to represent inequalities (e.g. $x \leq 10$)
    - Poor propagation strength
- **Attempt 2:** logarithmic encoding (create a boolean variable for each bit of x)
  - Logarithmic in size of domain, but even worse propagation strength

Order Encoding

- For each CP var x, create boolean variables:
  - $[\![x = i]\!]$ for $\mathrm{lb}(x) \leq i \leq \mathrm{ub}(x)$
  - $[\![x \leq i]\!]$ for $\mathrm{lb}(x) \leq i \leq \mathrm{ub}(x)$
- Note that $(x \geq i) \equiv \neg[\![x \leq i-1]\!]$ and $(x \neq i) \equiv \neg[\![x = i]\!]$
- Need to add consistency clauses:
  - $[\![x \leq i]\!] \Rightarrow [\![x \leq i+1]\!]$ for $\mathrm{lb}(x) \leq i \leq \mathrm{ub}(x) - 1$
  - $[\![x = i]\!] \Leftrightarrow [\![x \leq i]\!] \wedge \neg[\![x \leq i-1]\!]$
- Linear in size of domain; good propagation strength

Adding a CP constraint in SAT

- How can we write the constraint $x = y + z$ with clauses?
- Need to enforce it for each possible value of x, y, z
- For each $\mathrm{lb} \leq i, j \leq \mathrm{ub},$ add clauses:
  - $[\![y = i]\!] \land [\![z = j]\!] \Rightarrow [\![x = i + j]\!]$
  - $[\![x = i]\!] \land [\![z = j]\!] \Rightarrow [\![y = i - j]\!]$
  - $[\![x = i]\!] \land [\![y = j]\!] \Rightarrow [\![z = i - j]\!]$
- How many clauses? $O\left(\left|\mathrm{ub} - \mathrm{lb}\right|^2\right)$
  - What if we sum more variables? Exponential blowup!

Lazy Clause Generation

- **Key observation:** although it takes a lot of clauses to represent a CP constraint, most clauses are never used
- **Lazy clause generation:** rather than generate all these clauses before solving, just generate the ones we need, when we need them!
- OK, but how does that actually work...

Lazy Clause Generation

- Recall that FD propagators return an "explanation" for updating bounds, e.g. $(x \geq 4) \land (z \leq 2) \Rightarrow y \geq 2$
- Easy to express these explanations as clauses
- Can run propagators during execution of CDCL solver, then add explanation clauses to formula
    - If we only introduce explanation clauses when the LHS of the implication is currently true, they will immediately become unit clauses!

LCG Pseudocode

```
lazy_clause_generation(constraint_program):
    let P = make_propagators(constraint_program)
    if lcg_propagate() = CONFLICT: return INFEASIBLE
    while not all variables have been set:
        let x = pick_variable()
        create new decision level; set x = T
        while lcg_propagate(P,φ) = CONFLICT:
            if level = 0: return INFEASIBLE
            let (cls, lvl) = analyze_conflict()
            let φ = φ ∪ { cls }
            backjump(lvl)
    return FEASIBLE
```

```
lcg_propagate(P,φ):
  while True:
    if unit_prop() = CONFLICT:
      return CONFLICT
    for propagator p ∈ P:
      let expl_clauses = p(φ)
      let φ = φ ∪ expl_clauses
    if φ did not change:
      return SUCCESS
```

LCG Example

$$D_0(x_1) = D_0(x_2) = D_0(x_3) = D_0(x_4) = D_0(x_5) = [1..4]$$

s.t. $x_2 \leq x_5,\ \text{AllDifferent}([x_1, x_2, x_3, x_4]),\ x_1 + x_2 + x_3 + x_4 \leq 9$

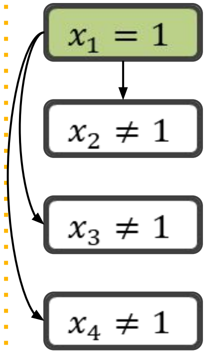
$\boxed{x_1 = 1}$

Decision: $[\![x_1 = 1]\!]$

(Note: For simplicity, some clauses are ignored in this example, and decision levels are left out; don't take it too seriously.)

LCG Example

AllDiff



Propagate $\text{AllDifferent}([x_1, x_2, x_3, x_4])$

Explanations: $x_1 = 1 \Rightarrow x_2 \neq 1$; $x_1 = 1 \Rightarrow x_3 \neq 1$; $x_1 = 1 \Rightarrow x_4 \neq 1$

LCG Example

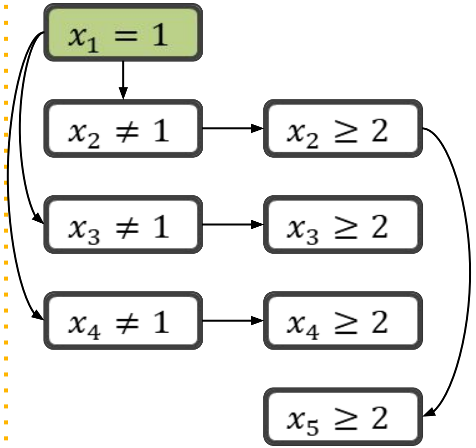
AllDiff

$$
\begin{tikzcd}
x_1 = 1 \arrow[d] \arrow[dd, bend right] \arrow[ddd, bend right] & \\
x_2 \neq 1 \arrow[r] & x_2 \geq 2 \\
x_3 \neq 1 \arrow[r] & x_3 \geq 2 \\
x_4 \neq 1 \arrow[r] & x_4 \geq 2
\end{tikzcd}
$$

Propagate consistency clauses

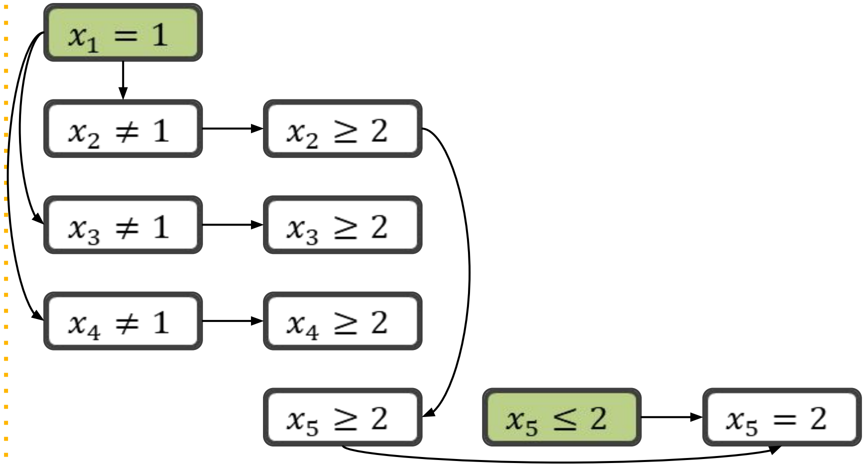
LCG Example

AllDiff $x_2 \leq x_5$



LCG Example

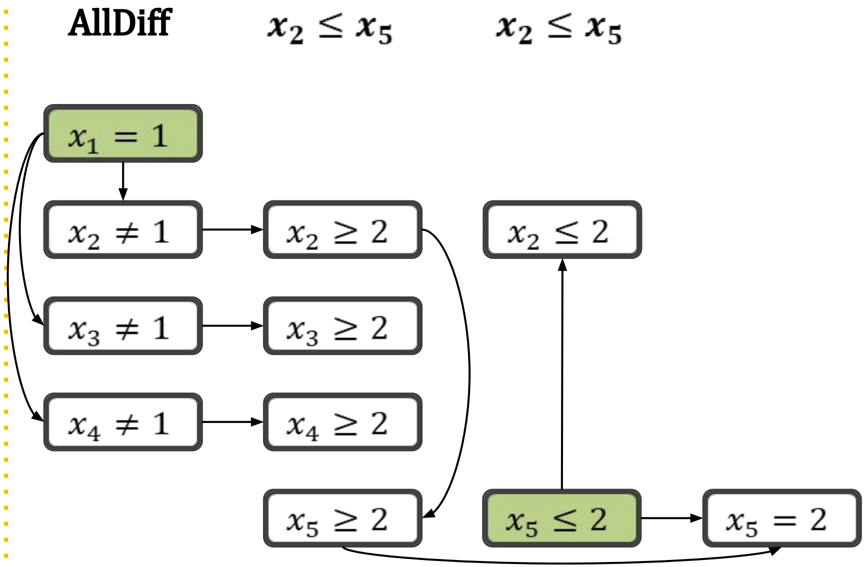
AllDiff $\boldsymbol{x_2 \leq x_5}$



Decision: $[\![x_5 \leq 2]\!]$

Propagate consistency constraints

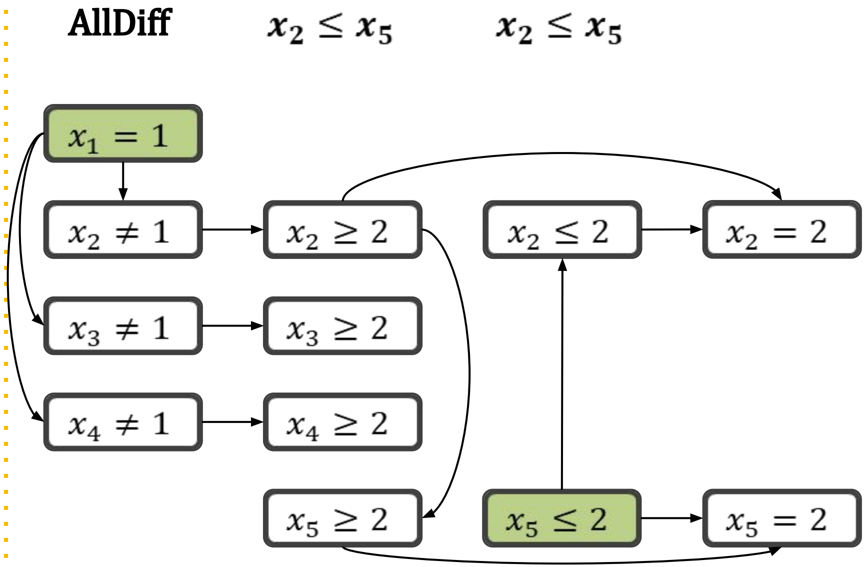
LCG Example



Propagate $x_2 \leq x_5$

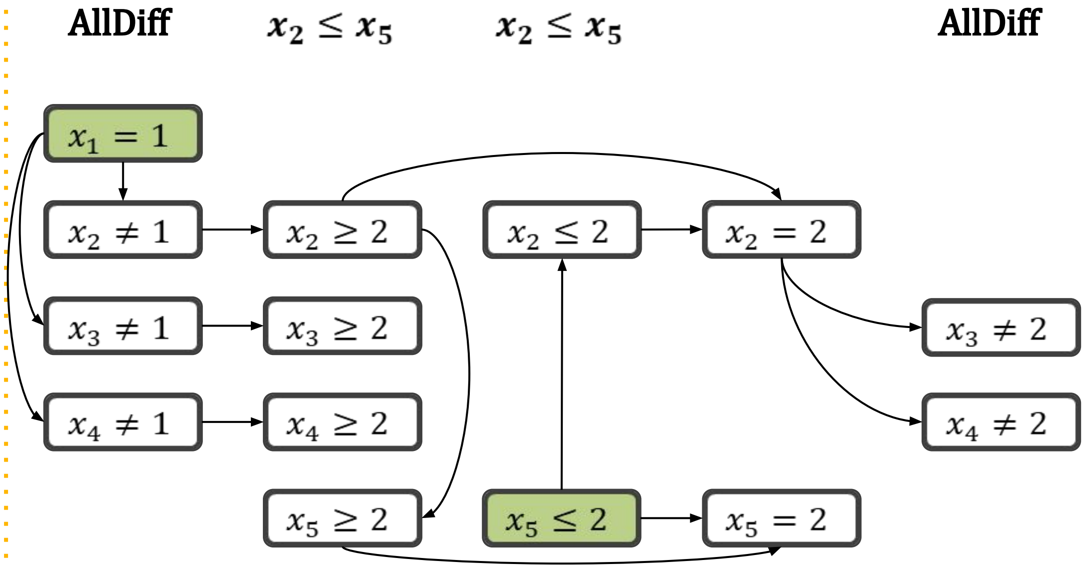
Explanations: $x_5 \leq 2 \Rightarrow x_2 \leq 2$

LCG Example



Propagate consistency constraints

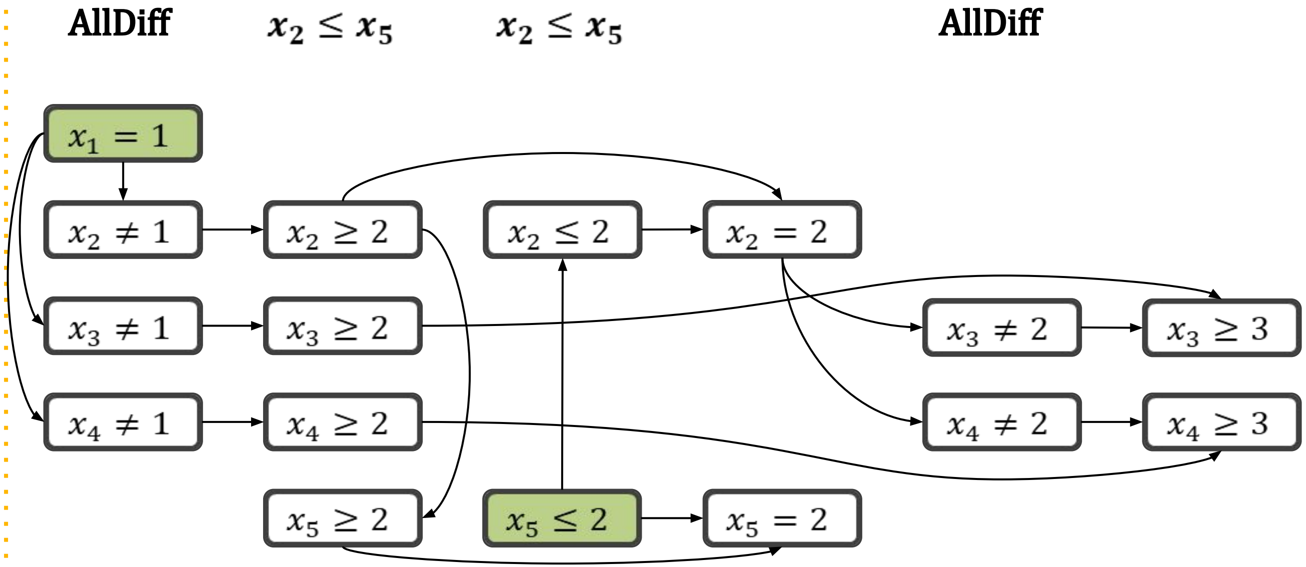
LCG Example



Propagate AllDifferent($[x_1, x_2, x_3, x_4]$)

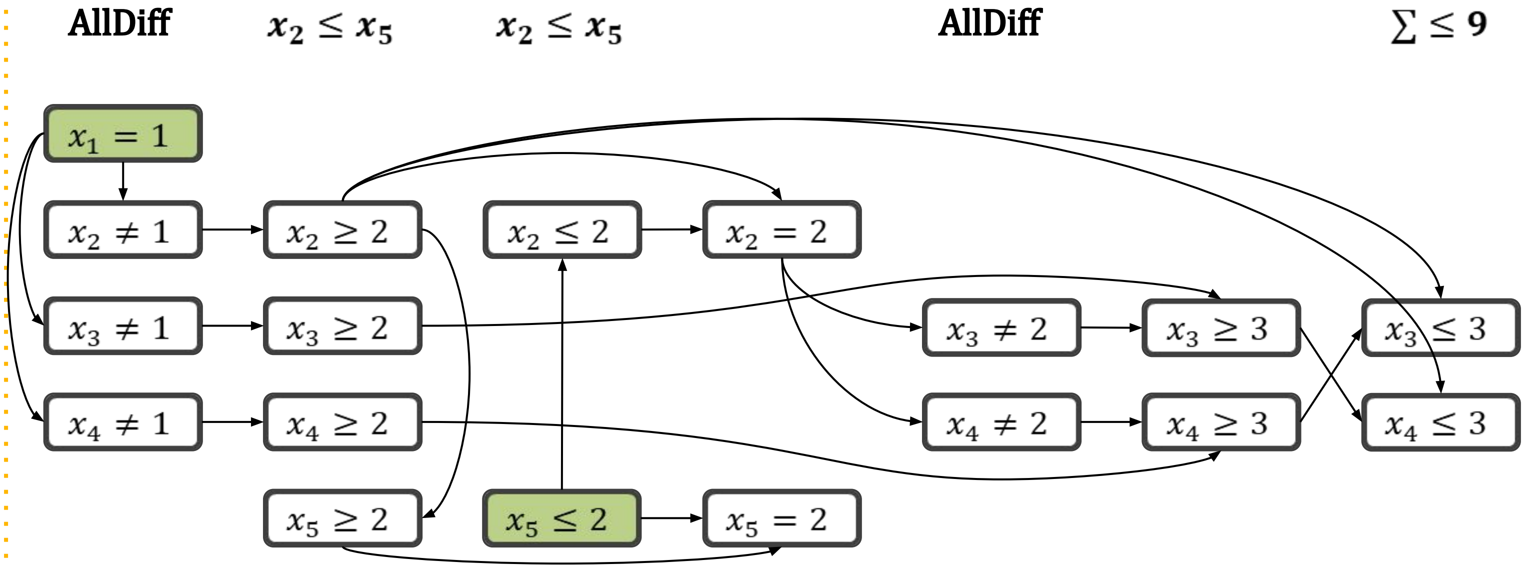
Explanations: $x_2 = 2 \Rightarrow x_3 \neq 2$; $x_2 = 2 \Rightarrow x_4 \neq 2$

LCG Example



Propagate consistency constraints

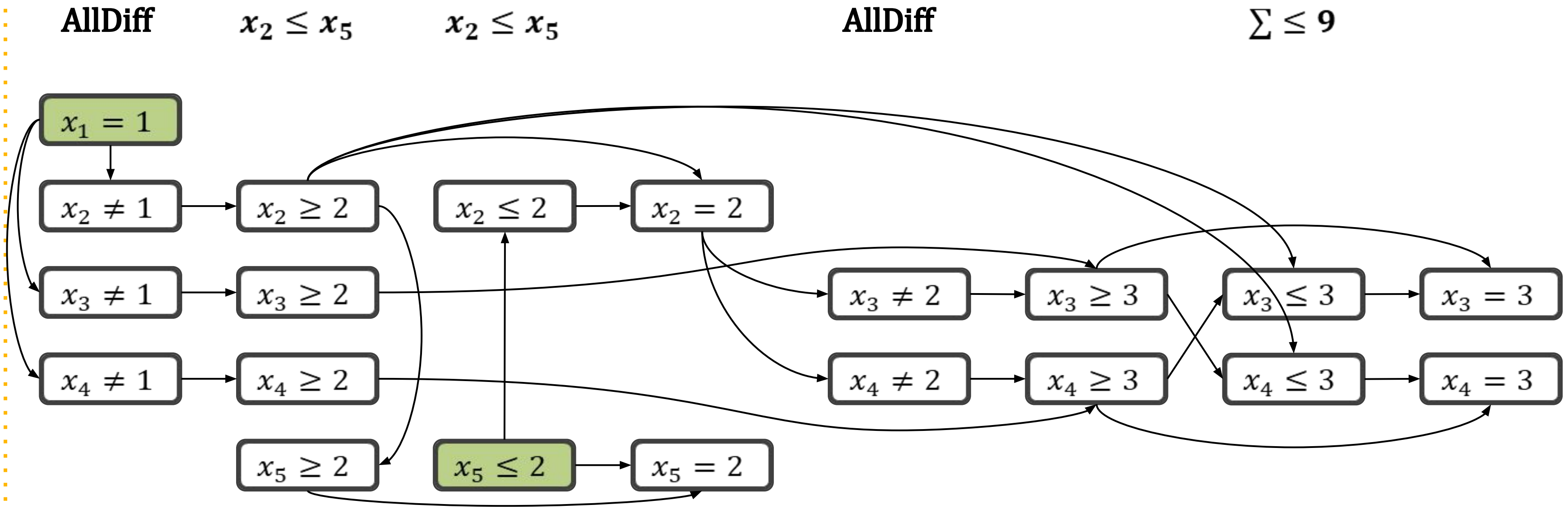
LCG Example



Propagate $x_1 + x_2 + x_3 + x_4 \le 9$

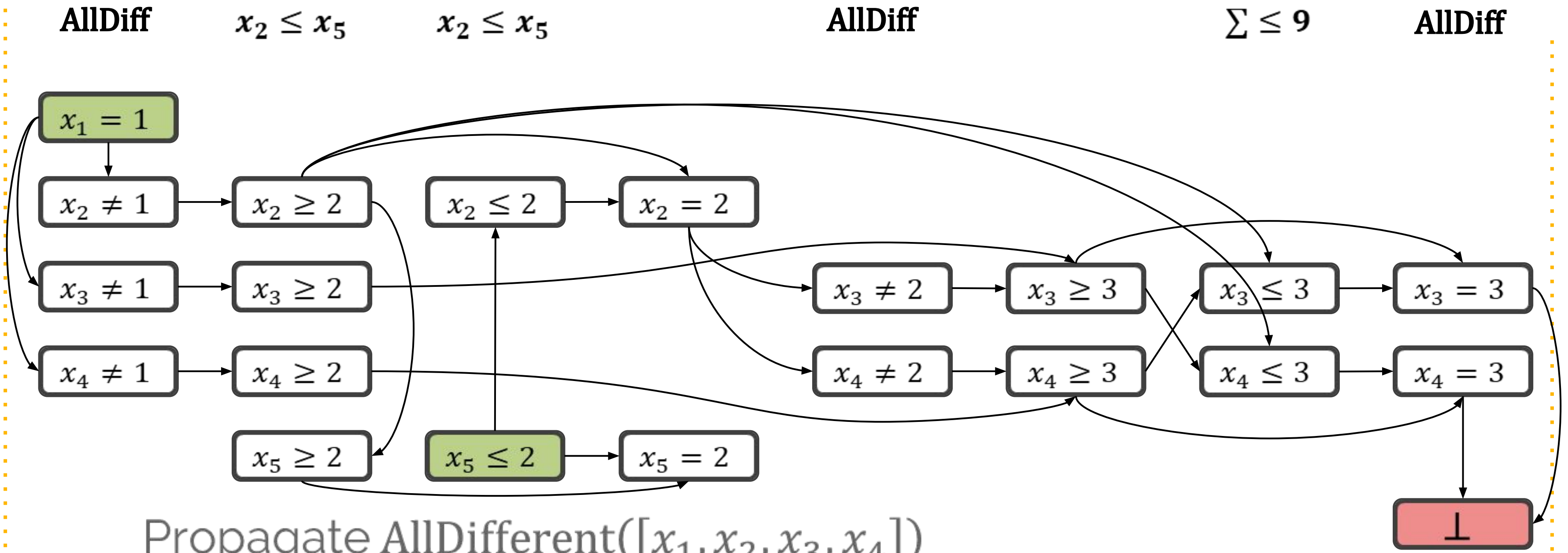
Explanations: $x_2 \ge 2 \wedge x_3 \ge 3 \Rightarrow x_4 \le 3$; $x_2 \ge 2 \wedge x_4 \ge 3 \Rightarrow x_3 \le 3$

LCG Example



Propagate consistency constraints

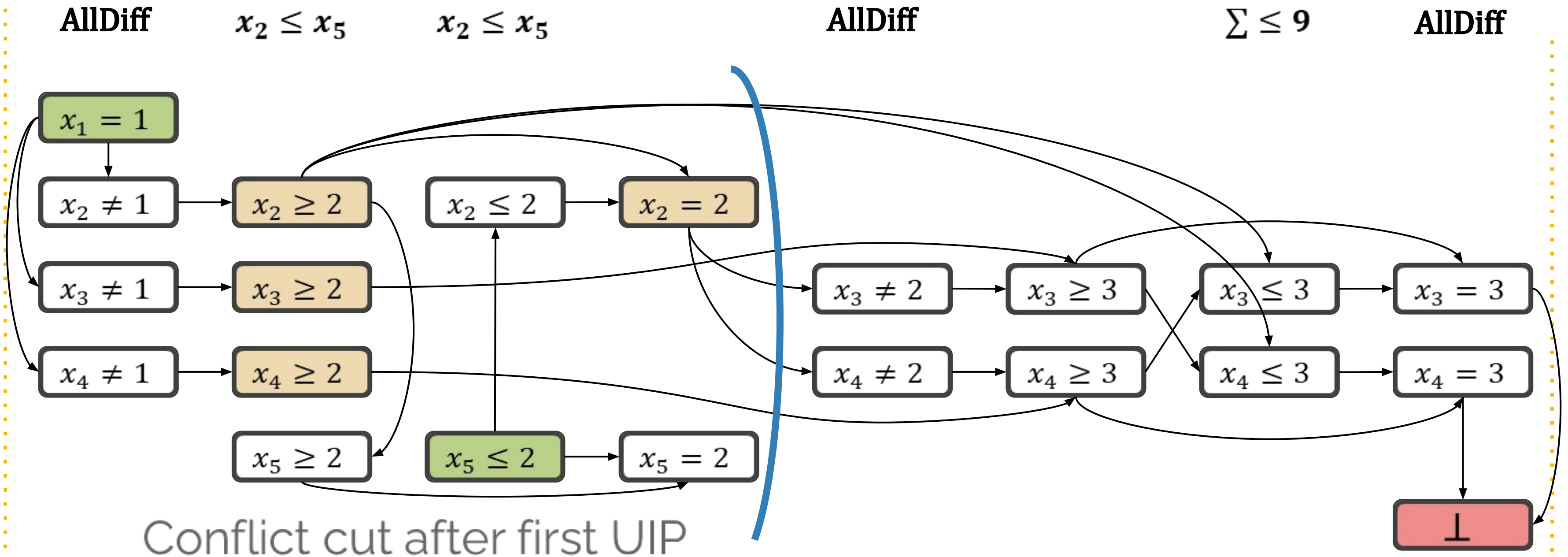
LCG Example



Propagate AllDifferent($[x_1, x_2, x_3, x_4]$)

Explanations: $x_3 = 3 \wedge x_4 = 3 \Rightarrow F$

LCG Example

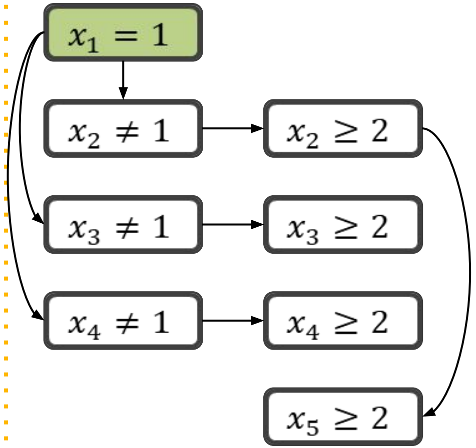


Conflict cut after first UIP

Learned clause: $\neg[\![x_2 \geq 2]\!] \vee \neg[\![x_3 \geq 2]\!] \vee \neg[\![x_4 \geq 2]\!] \vee \neg[\![x_2 = 2]\!]$

LCG Example

AllDiff $\boldsymbol{x_2 \le x_5}$

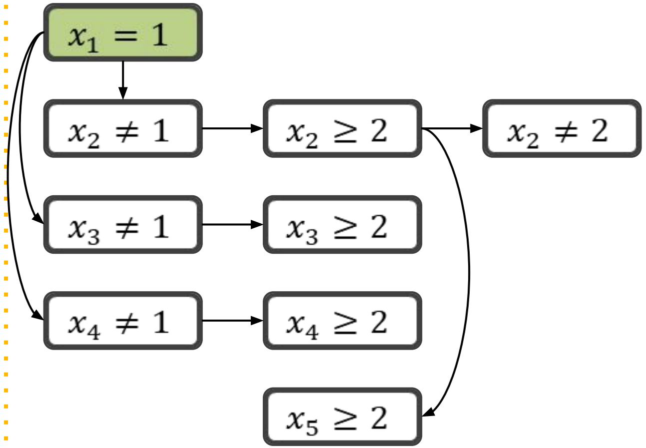


Backtrack to asserting level!

Learned clause: $\neg[\![x_2 \geq 2]\!] \vee \neg[\![x_3 \geq 2]\!] \vee \neg[\![x_4 \geq 2]\!] \vee \neg[\![x_2 = 2]\!]$

LCG Example

AllDiff $\quad \boldsymbol{x_2 \leq x_5}$



Propagate from learned clause

Learned clause: $\neg[\![x_2 \geq 2]\!] \vee \neg[\![x_3 \geq 2]\!] \vee \neg[\![x_4 \geq 2]\!] \vee \neg[\![x_2 = 2]\!]$

Explanation Deletion

- Explanation clauses are needed for immediate unit propagation and for generating learned clauses
- But when we backtrack past the explanations, may not need them anymore
    - Can delete from formula

Lazy Boolean Variable Creation

- Many of the boolean variables are never actually used
- **Idea:** create boolean variables when we need them
- **Array encoding:** initially only create $[\![x \leq i]\!]$ variables
  - Create $[\![x = i]\!]$ variables on demand
  - Don't forget to add clause: $[\![x = i]\!] \Leftrightarrow [\![x \leq i]\!] \land \neg[\![x \leq i - 1]\!]$
- **List encoding:** create both types of variables on demand!
  - When creating $[\![x \leq i]\!]$, add clauses:
    - $[\![x \leq i]\!] \Rightarrow [\![x \leq i_{\text{next}}]\!]$, where i_{next} is next-higher existing bnd

Lazy Variable Tradeoffs

- List encoding has fewer variables, so it can succeed on large domains where array encoding fails
- Array encoding interacts better with clause learning
    - This is significant!
- List encoding is trickier to implement

References

Feydy, T. J., & Stuckey, P. J. (2009). Lazy Clause Generation Reengineered. *Principles and Practice of Constraint Programming - CP 2009 Lecture Notes in Computer Science*, 352–366. doi: 10.1007/978-3-642-04244-7_29

Marriott, K., & Stuckey, P. J. (1999). *Programming with constraints: an introduction*. Cambridge (Massachusetts): MIT Press.

Ohrimenko, O., Stuckey, P. J., & Codish, M. (2009). Propagation via lazy clause generation. *Constraints*, *14*(3), 357–391. doi: 10.1007/s10601-008-9064-x

Stuckey, P. J. (2010). Lazy Clause Generation: Combining the Power of SAT and CP (and MIP?) Solving. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems Lecture Notes in Computer Science*, 5–9. doi: 10.1007/978-3-642-13520-0_3

Stuckey, P. J. (2010, June). Lazy Clause Generation: Combining the best of SAT and CP (and MIP?) solving. Retrieved from https://people.eng.unimelb.edu.au/pstuckey/cpaior-talk.pdf