Chapter 1: Constraints

What are they, what do they do and what can I use them for.

1

Constraints

- What are constraints?
- Modelling problems
- Constraint solving
- Tree constraints
- Other constraint domains
- Properties of constraint solving

2

Constraints

Variable: a place holder for values

$$X, Y, Z, L_3, U_{21}, List$$

Function Symbol: mapping of values to values

$$+, -, \times, \div, \sin, \cos, ||$$

Relation Symbol: relation between values

$$=, \leq, \neq$$

3

Constraints

Primitive Constraint: constraint relation with arguments

$$X \geq 4$$

$$X + 2Y = 9$$

Constraint: conjunction of primitive constraints

$$X \leq 3 \wedge X = Y \wedge Y \geq 4$$

4

Satisfiability

Valuation: an assignment of values to variables

$$\theta = \{X \mapsto 3, Y \mapsto 4, Z \mapsto 2\}$$

$$\theta(X + 2Y) = (3 + 2 \times 4) = 11$$

Solution: valuation which satisfies constraint

$$\theta(X \geq 3 \wedge Y = X + 1)$$

$$= (3 \geq 3 \wedge 4 = 3 + 1) = \mathit{true}$$

5

Satisfiability

Satisfiable: constraint has a solution

Unsatisfiable: constraint does not have a solution

$X \leq 3 \wedge Y = X + 1$ *satisfiable*

$X \leq 3 \wedge Y = X + 1 \wedge Y \geq 6$ *unsatisfiable*

6

Constraints as Syntax

- Constraints are strings of symbols
- Brackets don't matter (don't use them)
- $(X = 0 \wedge Y = 1) \wedge Z = 2 \;\equiv\; X = 0 \wedge (Y = 1 \wedge Z = 2)$
- Order does matter
- $X = 0 \wedge Y = 1 \wedge Z = 2 \;\not\equiv\; Y = 1 \wedge Z = 2 \wedge X = 0$
- Some algorithms will depend on order

7

Equivalent Constraints

Two different constraints can represent the same information

$$X > 0 \leftrightarrow 0 < X$$
$$X = 1 \wedge Y = 2 \leftrightarrow Y = 2 \wedge X = 1$$
$$X = Y + 1 \wedge Y \ge 2 \leftrightarrow X = Y + 1 \wedge X \ge 3$$

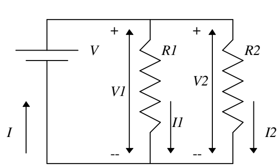
Two constraints are **equivalent** if they have the same set of solutions

8

Modelling with constraints

- Constraints describe idealized behaviour of objects in the real world

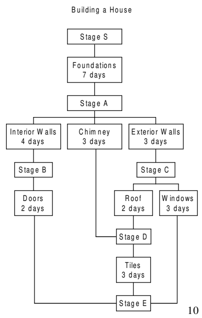
$$V1 = I1 \times R1$$
$$V2 = I2 \times R2$$
$$V - V1 = 0$$
$$V - V2 = 0$$
$$V1 - V2 = 0$$
$$I - I1 - I2 = 0$$
$$-I + I1 + I2 = 0$$



9

Modelling with constraints

start	$T_S \geq 0$
foundations	$T_A \geq T_S + 7$
interior walls	$T_B \geq T_A + 4$
exterior walls	$T_C \geq T_A + 3$
chimney	$T_D \geq T_A + 3$
roof	$T_D \geq T_C + 2$
doors	$T_E \geq T_B + 2$
tiles	$T_E \geq T_D + 3$
windows	$T_E \geq T_C + 3$



10

Constraint Satisfaction

- Given a constraint *C* two questions
  - **satisfaction**: does it have a solution?
  - **solution**: give me a solution, if it has one?
- The first is more basic
- A ***constraint solver*** answers the satisfaction problem

11

Constraint Satisfaction

- How do we answer the question?
- Simple approach try all valuations.

$X > Y$

$\{X \mapsto 1, Y \mapsto 1\}$ *false*
$\{X \mapsto 1, Y \mapsto 2\}$ *false*
$\{X \mapsto 1, Y \mapsto 3\}$ *false*
•
•
•

$X > Y$

$\{X \mapsto 1, Y \mapsto 1\}$ *false*
$\{X \mapsto 2, Y \mapsto 1\}$ *true*
$\{X \mapsto 2, Y \mapsto 2\}$ *false*
$\{X \mapsto 3, Y \mapsto 1\}$ *true*
$\{X \mapsto 3, Y \mapsto 2\}$ *true*
•
•

12

Constraint Satisfaction

- The enumeration method wont work for Reals (why not?)
- A smarter version will be used for finite domain constraints
- How do we solve Real constraints
- Remember Gauss-Jordan elimination from high school

13

Gauss-Jordan elimination

- Choose an equation *c* from *C*
- Rewrite *c* into the form *x = e*
- Replace *x* everywhere else in *C* by *e*
- Continue until
  - all equations are in the form *x = e*
  - or an equation is equivalent to *d = 0* (*d != 0*)
- Return *true* in the first case else *false*

14

Gauss-Jordan Example 1

$$\begin{array}{r} 1 + X = 2Y + Z \wedge \\ Z - X = 3 \wedge \\ X + Y = 5 + Z \end{array} \qquad 1 + X = 2Y + Z$$

Replace X by $2Y+Z-1$

$$\begin{array}{r} X = 2Y + Z - 1 \wedge \\ Z - 2Y - Z + 1 = 3 \wedge \\ 2Y + Z - 1 + Y = 5 + Z \end{array} \qquad -2Y = 2$$

Replace Y by -1

$$\begin{array}{r} X = -2 + Z - 1 \wedge \\ Y = -1 \wedge \\ -2 + Z - 1 - 1 = 5 + Z \end{array} \qquad -4 = 5$$

Return *false*

15

Gauss-Jordan Example 2

$$1 + X = 2Y + Z \wedge \quad 1+X=2Y+Z$$
$$Z - X = 3$$

Replace X by $2Y+Z-1$

$$X = 2Y + Z - 1 \wedge$$
$$Z - 2Y - Z + 1 = 3 \quad -2Y = 2$$

Replace Y by -1

$$X = Z - 3 \wedge$$
$$Y = -1$$

Solved form: constraints in this form are satisfiable

16

Solved Form

- **Non-parametric variable:** appears on the left of one equation.
- **Parametric variable:** appears on the right of any number of equations.
- **Solution:** choose parameter values and determine non-parameters

$$\begin{array}{l} X = Z - 3 \wedge \\ Y = -1 \end{array} \longrightarrow Z = 4 \longrightarrow \begin{array}{l} X = 4 - 3 = 1 \\ Y = -1 \end{array}$$

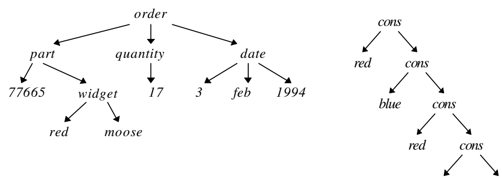
17

Tree Constraints

- Tree constraints represent structured data
- **Tree constructor:** character string
  - *cons, node, null, widget, f*
- **Constant:** constructor or number
- **Tree:**
  - A constant is a *tree*
  - A constructor with a list of > 0 trees is a *tree*
  - Drawn with constructor above *children*

18

Tree Examples



order(part(77665, widget(red, moose)), quantity(17), date(3, feb, 1994))

cons(red,cons(blue,cons(red,cons(…))))

19

Tree Constraints

- **Height of a tree:**
  - a constant has height 1
  - a tree with children *t1, …, tn* has height one more than the maximum of trees *t1,…,tn*
- **Finite tree:** has finite height
- Examples: height 4 and height ∞

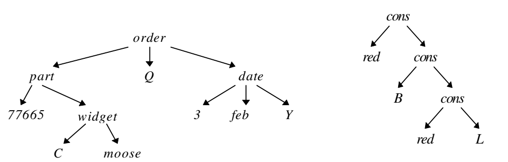
20

Terms

- A *term* is a tree with variables replacing subtrees
- **Term:**
  - A constant is a *term*
  - A variable is a *term*
  - A constructor with a list of > 0 terms is a *term*
  - Drawn with constructor above *children*
- **Term equation:** $s = t$ (s,t terms)

21

Term Examples



order(part(77665, widget(C, moose)), Q, date(3, feb, Y))

cons(red,cons(B,cons(red,L)))

22

Tree Constraint Solving

- Assign trees to variables so that the terms are identical
  - *cons(R, cons(B, nil)) = cons(red, L)*
  - $\{R \mapsto red, L \mapsto cons(blue, nil), B \mapsto blue\}$
- Similar to Gauss-Jordan
- Starts with a set of term equations *C* and an empty set of term equations *S*
- Continues until *C* is empty or it returns *false*

23

Tree Constraint Solving

- unify(C)
  - Remove equation *c* from *C*
  - **case** *x=x:* do nothing
  - **case** *f(s1,..,sn)=g(t1,..,tn):* **return** *false*
  - **case** *f(s1,..,sn)=f(t1,..,tn):*
    - add *s1=t1, .., sn=tn* to *C*
  - **case** *t=x* (*x* variable): add *x=t* to *C*
  - **case** *x=t* (*x* variable): add *x=t* to *S*
    - substitute *t* for *x* everywhere else in *C* and *S*

24

Tree Solving Example

C	S
$\underline{cons(Y,nil) = cons(X,Z)} \land Y = cons(a,T)$	$true$
$\underline{Y = X} \land nil = Z \land Y = cons(a,T)$	$true$
$\underline{nil = Z} \land X = cons(a,T)$	$Y = X$
$\underline{Z = nil} \land X = cons(a,T)$	$Y = X$
$\underline{X = cons(a,T)}$	$Y = X \land Z = nil$
$true$	$Y = cons(a,T) \land Z = nil \land X = cons(a,T)$

Like Gauss-Jordan, variables are parameters or non-parameters.
A solution results from setting parameters (I.e T) to any value.

$$\{T \mapsto nil, X \mapsto cons(a,nil), Y \mapsto cons(a,nil), Z \mapsto nil\}$$

25

One extra case

- Is there a solution to $X = f(X)$?
- NO!
  - if the height of X in the solution is n
  - then $f(X)$ has height $n+1$
- **Occurs check:**
  - before substituting t for x
  - check that x does not occur in t

26

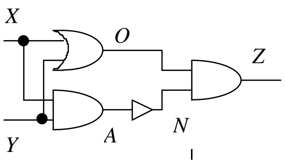
Other Constraint Domains

- There are many
  - Boolean constraints
  - Sequence constraints
  - Blocks world
- Many more, usually related to some well understood mathematical structure

27

Boolean Constraints

Used to model circuits, register allocation problems, etc.



$$O \leftrightarrow (X \vee Y) \wedge$$
$$A \leftrightarrow (X \& Y) \wedge$$
$$N \leftrightarrow \neg A \wedge$$
$$Z \leftrightarrow (O \& N)$$

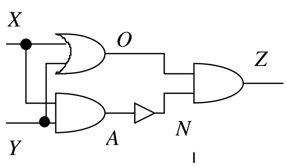
An exclusive or gate

Boolean constraint describing the xor circuit

28

Boolean Constraints

$$\neg FO \leftrightarrow (O \leftrightarrow (X \vee Y)) \wedge$$
$$\neg FA \leftrightarrow (A \leftrightarrow (X \& Y)) \wedge$$
$$\neg FN \leftrightarrow (N \leftrightarrow \neg A) \wedge$$
$$\neg FG \leftrightarrow (Z \leftrightarrow (N \& O))$$



Constraint modelling the circuit with faulty variables

$$\neg(FO \& FA) \wedge \neg(FO \& FN) \wedge \neg(FO \& FG) \wedge$$
$$\neg(FA \& FN) \wedge \neg(FA \& FG) \wedge \neg(FN \& FG)$$

Constraint modelling that only one gate is faulty

Observed behaviour: $\{X \mapsto 0, Y \mapsto 0, Z \mapsto 1\}$

Solution: $\{FO \mapsto 1, FA \mapsto 0, FN \mapsto 0, FG \mapsto 0,$
$X \mapsto 0, Y \mapsto 0, O \mapsto 1, A \mapsto 0, N \mapsto 1, Z \mapsto 1\}$

29

Boolean Solver

let m be the number of primitive constraints in C

$$n := \left\lceil \frac{\ln(\varepsilon)}{\ln(1-(1-\frac{1}{m})^m} \right\rceil$$ *epsilon is between 0 and 1 and determines the degree of incompleteness*

for i := 1 to n **do**

  generate a random valuation over the variables in C

  if the valuation satisfies C **then return** *true* **endif**

endfor

return *unknown*

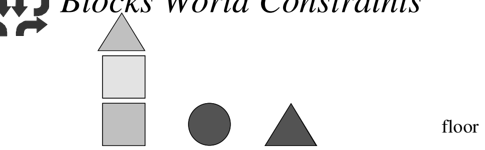
30

Boolean Constraints

- **Something new?**
- The Boolean solver can return *unknown*
- It is **incomplete** (doesnt answer all questions)
- It is polynomial time, where a complete solver is exponential (unless P = NP)
- Still such solvers can be useful!

31

Blocks World Constraints



Constraints don't have to be mathematical

Objects in the blocks world can be on the floor or on another object. Physics restricts which positions are stable. Primitive constraints are e.g. *red(X), on(X,Y), not_sphere(Y).*

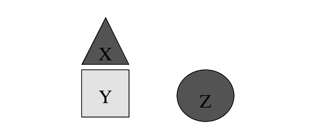
32

Blocks World Constraints

A solution to a Blocks World constraint is a picture

with an annotation of which variable is which block

$$yellow(Y) \wedge$$
$$red(X) \wedge$$
$$on(X,Y) \wedge$$
$$floor(Z) \wedge$$
$$red(Z)$$



33

Solver Definition

- A **constraint solver** is a function *solv* which takes a constraint *C* and returns *true*, *false* or *unknown* depending on whether the constraint is satisfiable
  - if $solv(C) = true$ then *C* is satisfiable
  - if $solv(C) = false$ then *C* is unsatisfiable

34

Properties of Solvers

- We desire solvers to have certain properties
- **well-behaved:**
  - **set based:** answer depends only on set of primitive constraints
  - **monotonic:** is solver fails for $C1$ it also fails for $C1 \wedge C2$
  - **variable name independent:** the solver gives the same answer regardless of names of vars

$$solv(X > Y \wedge Y > Z) = solv(T > U_1 \wedge U_1 > Z)$$

35

Properties of Solvers

- The most restrictive property we can ask
- **complete:** A solver is complete if it always answers *true* or *false*. (never *unknown*)

36

Constraints Summary

- Constraints are pieces of syntax used to model real world behaviour
- A constraint solver determines if a constraint has a solution
- Real arithmetic and tree constraints
- Properties of solver we expect (well-behavedness)

37