Constraint Logic Programming languages

Constraint Logic Programming (CLP)

Constraint Satisfaction Problems (CSP)

- artificial intelligence (1970s)
- e.g. $X \in \{1,2\} \wedge Y \in \{1,2\} \wedge Z \in \{1,2\} \wedge X = Y \wedge X \neq Z \wedge Y > Z$

Constraint Logic Programming (CLP)

- developed in the mid-1980s
- two declarative paradigms: constraint solving and logic programming
- more expressive, flexible, and in general more efficient than logic programs
- e.g. $X - Y = 3 \ \wedge \ X + Y = 7$ leads to $X = 5 \ \wedge \ Y = 2$
- e.g. $X < Y \ \wedge \ Y < X$ fails without the need to know values

Early history of constraint-based programming

1963	I. Sutherland, Sketchpad, graphic system for geometric drawing
1970	U. Montanari, Pisa, Constraint networks
1970	R.E. Fikes, REF-ARF, language for integer linear equations
1972	A. Colmerauer, U. Marseille, and R. Kowalski, IC London, Prolog
1977	A.K. Mackworth, Constraint networks algorithms
1978	J.-L. Lauriere, Alice, language for combinatorial problems
1979	A. Borning, Thinglab, interactive graphics
1980	G.L. Steele, Constraints, first constraint-based language, in LISP
1982	A. Colmerauer, Prolog II, U. Marseille, equality constraints
1984	Eclipse Prolog, ECRC Munich, later IC-PARC London
1985	SICStus Prolog, Swedish Institute of Computer Science (SICS)

Early history of constraint-based programming (cont)

1987	H. Ait-Kaci, U. Austin, Life, equality constraints
1987	J. Jaffar and J.L. Lassez, CLP(X) - Scheme, Monash U. Melbourne
1987	J. Jaffar, CLP($\Re$), Monash U. Melbourne, linear polynomials
1988	P. v. Hentenryck, CHIP, ECRC Munich, finite domains, Booleans
1988	P. Voda, Trilogy, Vancouver, integer arithmetics
1988	W. Older, BNR-Prolog, Bell-Northern Research Ottawa, intervals
1988	A. Aiba, CAL, ICOT Tokyo, non-linear equation systems
1988	W. Leler, Bertrand, term rewriting for defining constraints
1988	A. Colmerauer, Prolog III, U. Marseille, list constraints and more

CLP Syntax vs. LP Syntax

- signature augmented with *constraint symbols*
- consistent first-order constraint theory (CT)
- at least constraint symbols $\top$ and $\bot$
- syntactic equality $\doteq$ as constraints (by including CET into CT)
- constraints handled by predefined, given constraint solver

CLP Syntax (cont)

- *atom*: expression $p(t_1, \ldots, t_n)$, with predicate symbol p/n
- *atomic constraint*: expression $c(t_1, \ldots, t_n)$, with n-ary constraint symbol c/n
- *constraint*:
  - atomic constraint, or
  - conjunction of constraints
- *goal*:
  - $\top$ (top), or $\bot$ (bottom), or
  - atom, or an atomic constraint, or
  - conjunction of goals
- *(CL) clause*: $A \leftarrow G$, with atom A and goal G
- *CL program*: finite set of CL clauses

CLP Syntax – Summary

Atom:	A, B	$::=$	$p(t_1, \ldots, t_n),\ n \geq 0$
Constraint:	C, D	$::=$	$c(t_1, \ldots, t_n) \mid C \wedge D,\ n \geq 0$
Goal:	G, H	$::=$	$\top \mid \bot \mid A \mid C \mid G \wedge H$
CL Clause:	K	$::=$	$A \leftarrow G$
CL Program:	P	$::=$	$K_1 \ldots K_m,\ m \geq 0$

CLP State Transition System

- *state* $\langle G, C \rangle$:G goal (store), C constraint (store)
- *initial state*: $\langle G,\top \rangle$
- *successful final state*: $\langle \top,\mathrm{C} \rangle$ and C is different from $\perp$
- *failed final state*: $\langle G, \perp \rangle$
- *successful and failed derivations and goals*: as in LP calculus

CLP Operational Semantics

Unfold

If	$(B \leftarrow H)$ is a fresh variant of a clause in P
and	$CT \models \exists\, ((B \doteq A) \wedge C)$
then	$\langle A \wedge G, C \,\rangle \quad \mapsto \langle H \wedge G, (B \doteq A) \wedge C \,\rangle$

Failure

If	there is no clause $(B \leftarrow H)$ in P
with	$CT \models \exists\, ((B \doteq A) \wedge C)$
then	$\langle A \wedge G, C \,\rangle \quad \mapsto \langle\, \perp, \perp \,\rangle$

Solve

If	$CT \models \forall\, ((C \wedge D_1) \leftrightarrow D_2)$
then	$\langle C \wedge G, D_1 \,\rangle \quad \mapsto \langle G, D_2 \,\rangle$

CLP Unfold – Comparison with LP

Unfold	
If	$(B \leftarrow H)$ is a fresh variant of a clause in P
and	$CT \models \exists\, ((B \doteq A) \wedge C)$
then	$\langle A \wedge G, C\ \rangle \quad \mapsto \langle H \wedge G, (B \doteq A) \wedge C\ \rangle$

- generalization of LP
- most general unifier in LP
- equality constraint between B and A in context of constraint store C, add equality constraint to store C

($(B \doteq A)$: shorthand for equating arguments of B and A pairwise)

CLP Solve

Solve

If $CT \models \forall\, ((C \wedge D_1) \leftrightarrow D_2)$

then $\langle C \wedge G, D_1 \rangle \mapsto \langle G, D_2 \rangle$

- form of simplification depends on constraint system and its constraint solver
- trying to simplify inconsistent constraints to $\perp$
- a failed final state can be reached using **Solve**

CLP State Transition System (vs. LP)

- like in LP, two degrees of non-determinism in the calculus (selecting the goal and selecting the clause)
- like in LP, search trees (mostly SLD resolution)
- *LP*: accumulate and compose substitutions
  CLP: accumulate and simplify constraints
- like substitutions, constraints never removed from constraint store (information increases monotonically during derivations)

CLP as Extension to LP

- derivation in LP can be expressed as CLP derivations:
  - *LP*: substitution $\{X_1 \mapsto t_1, \ldots, X_n \mapsto t_n\}$
  - *CLP*: equality constraints: $X_1 \doteq t_1 \wedge \ldots \wedge X_n \doteq t_n$
- CLP generalizes form of answers.
  - *LP answer*: substitution
  - *CLP answer*: constraint
- Constraints summarize several (even infinitely many) LP answers into one (intensional) answer, e.g.,
  - $X+Y \geq 3 \;\wedge\; X+Y \leq 3$ simplified to
  - $X+Y \doteq 3$
  - variables do not need to have a value

CLP Logical Reading, Answer Constraint

- *logical reading of a state* $\langle H, C\ \rangle$: $\exists\bar{X}(H \wedge C)$
  - $\langle G,\top\ \rangle \quad \mapsto^{*}\langle H, C\ \rangle$
  - $\bar{X}$: variables which occur in H or C but not in G
- *answer (constraint) of a goal* G:
  logical reading of final state of derivation starting with $\langle G,\top\ \rangle$

Answer constraints of final states:

- $\langle \top, C\rangle$ is true as $(\exists\bar{X}\top \wedge C) \Leftrightarrow \exists\bar{X}C$
- $\langle G, \bot\rangle$ is false as $(\exists\bar{X}G \wedge \bot) \Leftrightarrow \bot$

CLP Example – Min

$\mathrm{min(X,Y,Z)} \leftarrow \mathrm{X} \leq \mathrm{Y} \wedge \mathrm{X} \doteq \mathrm{Z}$ $(c1)$
$\mathrm{min(X,Y,Z)} \leftarrow \mathrm{Y} \leq \mathrm{X} \wedge \mathrm{Y} \doteq \mathrm{Z}$ $(c2)$
Constraints with usual meaning

- $\leq$ total order
- $\doteq$ syntactic equality

CLP Example – Search Tree for `min(1,2,C)`

$$@C - 16ex$$

$$\langle \mathtt{min}(1,2,\mathtt{C}),\ \mathtt{true}\rangle \quad [\mathrm{dl}]\ [\mathrm{dr}]$$
$$\langle \mathtt{X}{\le}\mathtt{Y} \wedge \mathtt{X}{\doteq}\mathtt{Z},\ 1{\doteq}\mathtt{X} \wedge 2{\doteq}\mathtt{Y} \wedge \mathtt{C}{\doteq}\mathtt{Z}\rangle \quad [\mathrm{d}]$$
$$\langle \mathtt{Y}{\le}\mathtt{X} \wedge \mathtt{Y}{\doteq}\mathtt{Z},\ 1{\doteq}\mathtt{X} \wedge 2{\doteq}\mathtt{Y} \wedge \mathtt{C}{\doteq}\mathtt{Z}\rangle \quad [\mathrm{d}]$$
$$\langle \top,\ \mathtt{C}{\doteq}\ 1\rangle$$

Goal `min(1,2,C)`:

$$\begin{array}{ll} & \langle \mathtt{min}(1,2,\mathtt{C}),\ \mathtt{true}\ \rangle \\ \mapsto_{\textbf{Unfold}\ (c1)} & \langle \mathtt{X}{\le}\mathtt{Y} \wedge \mathtt{X}{\doteq}\mathtt{Z},\quad 1{\doteq}\mathtt{X}\ \wedge 2{\doteq}\mathtt{Y} \wedge \mathtt{C}{\doteq}\mathtt{Z}\rangle \\ \mapsto_{\textbf{Solve}} & \langle \top,\ \mathtt{C}{\doteq}1\rangle \end{array}$$

Using $(c2)$ leads to inconsistent constraint store $2{\le}1 \wedge 2{\doteq}\mathtt{C}$ – derivation fails

CLP Example – Min (More Derivations)

min(X,Y,Z) $\leftarrow$ X$\leq$Y $\wedge$ X$\doteq$Z $(c1)$
min(X,Y,Z) $\leftarrow$ Y$\leq$X $\wedge$ Y$\doteq$Z $(c2)$

- Goal `min(A,2,1)`:

$$\begin{array}{ll} & \langle \texttt{min}(\texttt{A},2,1),\ \texttt{true}\ \rangle \\ \mapsto_{\textbf{Unfold (c1)}} & \langle \texttt{X}\leq\texttt{Y}\ \wedge\ \texttt{X}\doteq\texttt{Z},\quad \texttt{A}\doteq\texttt{X}\ \wedge\ 2\doteq\texttt{Y}\ \wedge\ 1\doteq\texttt{Z}\rangle \\ \mapsto_{\textbf{Solve}} & \langle \top,\ \texttt{A}\doteq 1\rangle \end{array}$$

  but fails with $(c2)$.
- `min(A,2,2)` has answer A$\doteq$2 for $(c1)$, and $2 \leq A$ for $(c2)$
- `min(A,2,3)` fails

(In Prolog, these transitions would lead to error messages.)

CLP Example – Min (More Derivations 2)

$\mathrm{min}(X,Y,Z) \leftarrow X \leq Y \wedge X \doteq Z$ ($c1$)
$\mathrm{min}(X,Y,Z) \leftarrow Y \leq X \wedge Y \doteq Z$ ($c2$)

- `min(A,A,B)` using ($c1$) (same answer with ($c2$))

$$
\begin{array}{ll}
 & \langle \mathtt{min}(\mathtt{A},\mathtt{A},\mathtt{B}),\ \mathtt{true}\ \rangle \\
\mapsto_{\textbf{Unfold (c1)}} & \langle \mathtt{X} \leq \mathtt{Y}\ \wedge\ \mathtt{X} \doteq \mathtt{Z},\quad \mathtt{A} \doteq \mathtt{X}\ \wedge\ \mathtt{A} \doteq \mathtt{Y}\ \wedge\ \mathtt{B} \doteq \mathtt{Z} \rangle \\
\mapsto_{\textbf{Solve}} & \langle \top,\ \mathtt{A} \doteq \mathtt{B} \rangle
\end{array}
$$

- General goal `min(A,B,C)` $\wedge$ $\mathtt{A} \leq \mathtt{B}$
  - using ($c1$): answer $\mathtt{A} \doteq \mathtt{C}\ \wedge\ \mathtt{A} \leq \mathtt{B}$
  - using ($c2$): answer $\mathtt{A} \doteq \mathtt{C}\ \wedge\ \mathtt{A} \doteq \mathtt{B}$ (more specific)
- `min(A,B,C)` ?

CLP Example – Min (Logical Reading)

min(X,Y,Z) $\leftarrow$ X$\leq$Y $\wedge$ X$\doteq$Z ($c1$)
min(X,Y,Z) $\leftarrow$ Y$\leq$X $\wedge$ Y$\doteq$Z ($c2$)

$$\forall X_1 X_2 X_3 \texttt{min}(X_1, X_2, X_3) \leftrightarrow$$
$$(\exists Y_{11} Y_{12} Y_{13}\ Y_{11} \doteq X_1,\ Y_{12} \doteq X_2,\ Y_{13} \doteq X_3 \wedge Y_{11} \leq Y_{12} \wedge Y_{11} \doteq Y_{13}$$
$$(\exists Y_{21} Y_{22} Y_{23}\ Y_{21} \doteq X_1,\ Y_{22} \doteq X_2,\ Y_{23} \doteq X_3 \wedge Y_{22} \leq Y_{21} \wedge Y_{22} \doteq Y_{23}$$

In shorthand:

$$\forall X \left(\texttt{min}(X_1, X_2, X_3) \leftrightarrow (X_1 \leq X_2 \wedge X_1 \doteq X_3) \vee (X_2 \leq X_1 \wedge X_2 \doteq X_3)\right)$$

CLP Declarative Semantics of P

Union of $P^{\leftrightarrow}$ with a *constraint theory* CT
(in LP only the special theory theory CET used)

- **Soundness:**
  If G has successful derivation with answer constraint C, then $P^{\leftrightarrow} \cup CT \models \forall(C \rightarrow G)$.
- **Completeness:**
  If $P^{\leftrightarrow} \cup CT \models \forall(C \rightarrow G)$ and C is satisfiable in CT, then there are successful derivations for G with answer constraints $C_1, \ldots, C_n$ s.t. $CT \models \forall(C \rightarrow (C_1 \vee \ldots \vee C_n))$.

(P CL program, G goal)

CLP Example – Completeness

P p(X,Y) $\leftarrow$ X$\leq$Y
p(X,Y) $\leftarrow$ X$\geq$Y

$P^{\leftrightarrow}$
$\forall X \forall Y p(X, Y) \leftrightarrow (X \leq Y \vee Y \leq X)$
(CT total order $\leq$)

Completeness:
As $P^{\leftrightarrow} \cup CT \models \forall(\mathtt{true} \rightarrow \mathtt{p}(\mathtt{X}, \mathtt{Y}))$ there are successful derivations for the goal `p(X,Y)`. The answer constraints $\mathtt{X}\leq\mathtt{Y}$ and $\mathtt{X}\geq\mathtt{Y}$ of `p(X,Y)` satisfy $CT \models \forall(\mathtt{true} \rightarrow \mathtt{X}\leq\mathtt{Y} \vee \mathtt{X}\geq\mathtt{Y})$.

But: Each answer on its own is not sufficient:
$CT \not\models \forall(\mathtt{true} \rightarrow \mathtt{X}\leq\mathtt{Y})$ and $CT \not\models \forall(\mathtt{true} \rightarrow \mathtt{X}\geq\mathtt{Y})$.

CLP Failed derivations

Soundness and Completeness:
$P^{\leftrightarrow} \cup CT \models \neg\exists G$ if and only if
each fair derivation starting with $\langle G, \top \rangle$ fails finitely
(P, CL program, G goal)

CLP Stability Property (Monotonicity)

If

- $\langle G, C \mapsto\rangle \ _{\textbf{Unfold}}\langle G', C' \ \rangle$
- $C' \wedge D$ satisfiable
  (ensures correctness of computation step in any larger context)

then also $\langle G \wedge H, C \wedge D \mapsto\rangle \ _{\textbf{Unfold}}\langle G' \wedge H, C' \wedge D \ \rangle$.

(D constraint, H goal)

CLP vs. LP – Overview

- *generate-and-test in LP*: impractical, facts used in passive manner only
- *constrain-and-generate in CLP*: use facts in active manner to reduce the search space (constraints)
- combination of
  - *LP languages*: declarative, for arbitrary predicates, non-deterministic
  - *constraint solvers*: declarative, efficient for special predicates, deterministic
- combination of search with constraints solving particularly useful

CLP Constrain/Generate vs. LP Generate/Test

Crypto-arithmetic Puzzle – Send More Money

$$\begin{array}{cccccc} & & S & E & N & D \\ + & & M & O & R & E \\ \hline = & M & O & N & E & Y \end{array}$$

Replace distinct letters by distinct digits, numbers have no leading zeros.

CLP Example – Send More Money (Solution)

```
[S,E,N,D,M,O,R,Y] = [9,5,6,7,1,0,8,2]
```

$$
\begin{array}{cccccc}
 & & S & E & N & D \\
 & & 9 & 5 & 6 & 7 \\
 & & M & O & R & E \\
+ & & 1 & 0 & 8 & 5 \\
\hline
 & M & O & N & E & Y \\
= & 1 & 0 & 6 & 5 & 2
\end{array}
$$

CLP Example – Send More Money (Constrain/Generate)

```
:- use_module(library(clpfd)).

send([S,E,N,D,M,O,R,Y]) :-
     gen_domains([S,E,N,D,M,O,R,Y],0..9),
     S #\= 0, M #\= 0,
     all_distinct([S,E,N,D,M,O,R,Y]),
                 1000*S + 100*E + 10*N + D
     +           1000*M + 100*O + 10*R + E
     #= 10000*M + 1000*O + 100*N + 10*E + Y,
     labeling([],[S,E,N,D,M,O,R,Y]).

gen_domains([],_).
gen_domains([H|T],D) :- H in D, gen_domains(T,D).
```

CLP Example – Send More Money (Constrain/Generate 2)

`send` **Without labeling**

```
:- send([S,E,N,D,M,O,R,Y]).
M = 1, O = 0, S = 9,
E in 4..7,
N in 5..8,
D in 2..8,
R in 2..8,
Y in 2..8 ?
```

		S	E	N	D
+		M	O	R	E
=	M	O	N	E	Y

CLP Example – Send More Money (Constrain/Generate 3)

`send` **Without labeling**

```
:- send([9,4,N,D,M,O,R,Y]).
no
```

Propagation determines `N = 5`, `R = 8`, but fails as `D` has no possible value. But

```
:- send([9,5,N,D,M,O,R,Y]).
D = 7, M = 1, N = 6,
O = 0, R = 8, Y = 2
yes
```

already computes solution.

		S	E	N	D
+		M	O	R	E
=	M	O	N	E	Y

LP Example – Send More Money (Generate/Test)

```
send([S,E,N,D,M,O,R,Y]) :-
     gen_domains([S,E,N,D,M,O,R,Y],0..9),
     labeling([],[S,E,N,D,M,O,R,Y]),
     S #\= 0, M #\= 0,
     all_distinct([S,E,N,D,M,O,R,Y]),
                  1000*S + 100*E + 10*N + D
     +            1000*M + 100*O + 10*R + E
     #= 10000*M + 1000*O + 100*N + 10*E + Y.
```

95,671,082 choices to find the solution

Houses logical puzzle. Folklore attributes this puzzle to Einstein

- Five colored houses in a row, each with an owner, a pet, cigarettes, and a drink.
- Each house has a different color
- Each owner has a different nationality
- Each owner has a different pet
- Each owner smoke a different brand of cigarette
- Each owner has a different drink

Plus the following contsraints:

Houses logical puzzle: constraints

1. The English lives in the red house.
2. The Spanish has a dog.
3. They drink coffee in the green house.
4. The Ukrainian drinks tea.
5. The green house is next to the white house.
6. The Winston smoker has a serpent.
7. In the yellow house they smoke Kool.
8. In the middle house they drink milk.
9. The Norwegian lives in the first house from the left.
10. The Chesterfield smoker lives near the man with the fox.
11. In the house near the house with the horse they smoke Kool. Lucky Strike smoker drinks juice. Japanese smokes Kent.
12. The Norwegian lives near the blue house.

Who owns the zebra and who drinks water?

A Prolog solution

```
houses(Hs) :-
% each house in the list Hs of houses is represented as:
%  h(Nationality, Pet, Cigarette, Drink, Color)}
      length(Hs, 5), % 1
      member(h(english,_,_,_,red), Hs),  % 2
      member(h(spanish,dog,_,_,_), Hs), % 3
      member(h(_,_,_,coffee,green), Hs), % 4
      member(h(ukrainian,_,_,tea,_), Hs), % 5
      next(h(_,_,_,_,green), h(_,_,_,_,white), Hs), % 6
      member(h(_,snake,winston,_,_), Hs),  % 7
      member(h(_,_,kool,_,yellow), Hs),  % 8
      Hs = [_,_,h(_,_,_,milk,_),_,_],  % 9
      Hs = [h(norwegian,_,_,_,_)|_],  %10
      next(h(_,fox,_,_,_), h(_,_,chesterfield,_,_), Hs), %
      next(h(_,_,kool,_,_), h(_,horse,_,_,_), Hs),  %12
      member(h(_,_,lucky,juice,_), Hs),  %13
      member(h(japanese,_,kent,_,_), Hs),  %14
      next(h(norwegian,_,_,_,_), h(_,_,_,_,blue), Hs), %15
      member(h(_,_,_,water,_), Hs), %one of them drinks
                                              %water
      member(h(_,zebra,_,_,_), Hs). %one of them owns
                                              % a zebra
```

Solution (ctnd)

```
zebra_owner(Owner) :-
        houses(Hs),
        member(h(Owner,zebra,_,_,_), Hs).

water_drinker(Drinker) :-
        houses(Hs),
        member(h(Drinker,_,_,water,_), Hs).

next(A, B, Ls) :- append(_, [A,B|_], Ls).
next(A, B, Ls) :- append(_, [B,A|_], Ls).
```

Examples of goals (queries:)

```
?- zebra_owner(Owner).
?- water_drinker(Drinker).
?- houses(Houses).
```

A CLP(FD) Solution

1. 25 Variables:
   - nationality: english, spaniard, japanese, italian, norwegian,
   - pet: dog, snails, fox, horse, zebra,
   - profession: painter, sculptor, diplomat, violinist, doctor,
   - drink: tea, coffee, milk, juice, water,
   - colour: red, green, white, yellow, blue.
2. Domain: [1...5].
3. Constraints:

```
alldifferent(red,green,white,yellow,blue),
alldifferent(english,spaniard,japanese,italian,norwegian
alldifferent(dog,snails,fox,horse,zebra),
alldifferent(painter,sculptor,diplomat,violinist,doctor)
alldifferent(tea,coffee,milk,juice,water).
```

Constrains (ctnd)

```
% The Englishman lives in the red house:
english = red,
% The Spaniard has a dog:
spaniard = dog,
% The Japanese is a painter:
japanese = painter,
% The Italian drinks tea:
italian = tea,
% The Norwegian lives in the first house on
the left:
norwegian = 1,
& The owner of the green house drinks coffee:
green = coffee,
% The green house is on the right of the white
house:
green = white + 1,
% The sculptor breeds snails:
sculptor = snails,
%  The diplomat lives in the yellow house:
diplomat = yellow,
```

Constrains (ctnd)

```
%  They drink milk in the middle house:
milk = 3,
%  The Norwegian lives next door to the blue house:
| norwegian ? blue | = 1,
% The violinist drinks fruit juice:
violinist = juice,
% The fox is in the house next to the doctor's:
|fox ? doctor| = 1,
% The horse is in the house next to the diplomat's:
| horse ? diplomat | = 1.
```

Exercise: complete the program by using these constraints.