Constraint Programming with CHR

Thom Frühwirth

Faculty of Computer Science
University of Ulm, Germany

Uni Dortmund
November 2005

Citations may be incomplete for space reasons

Constraint Reasoning and Programming

Part I

Constraint Programming

The Holy Grail

Constraint Programming represents one of the closest approaches computer science has yet made to the **Holy Grail** of programming: the user states the problem, the computer solves it.

Eugene C. Freuder, Inaugural issue of the *Constraints Journal*, 1997.

Constraint Reasoning

The Idea

- *Combination Lock Example*
  0 1 2 3 4 5 6 7 8 9
  Greater or equal 5.
  Prime number.
- *Declarative problem* representation by variables and constraints:
  $x \in \{0, 1, \ldots, 9\} \;\wedge\; x \geq 5 \;\wedge\; \mathrm{prime}(x)$
- *Constraint propagation and simplification* reduce search space:
  $x \in \{0, 1, \ldots, 9\} \wedge x \geq 5 \;\rightarrow\; x \in \{5, 6, 7, 8, 9\}$

Constraint Reasoning

The Idea

- *Combination Lock Example*
  $\cancel{0}$ $\cancel{1}$ $\cancel{2}$ $\cancel{3}$ $\cancel{4}$ 5 6 7 8 9
  Greater or equal 5.
  Prime number.
- *Declarative problem* representation by variables and constraints:
  $x \in \{0, 1, \ldots, 9\} \ \wedge \ x \geq 5 \ \wedge \ \text{prime}(x)$
- *Constraint propagation and simplification* reduce search space:
  $x \in \{0, 1, \ldots, 9\} \wedge x \geq 5 \ \rightarrow \ x \in \{5, 6, 7, 8, 9\}$

Constraint Reasoning

The Idea

- *Combination Lock Example*
  $\cancel{0}$ $\cancel{1}$ $\cancel{2}$ $\cancel{3}$ $\cancel{4}$ 5 $\cancel{6}$ 7 $\cancel{8}$ $\cancel{9}$
  Greater or equal 5.
  Prime number.
- *Declarative problem* representation by variables and constraints:
  $x \in \{0, 1, \ldots, 9\} \ \wedge \ x \geq 5 \ \wedge \ \text{prime}(x)$
- *Constraint propagation and simplification* reduce search space:
  $x \in \{0, 1, \ldots, 9\} \wedge x \geq 5 \ \rightarrow \ x \in \{5, 6, 7, 8, 9\}$

Constraint Reasoning Everywhere

Combination

Simplification

Contradiction

Redundancy

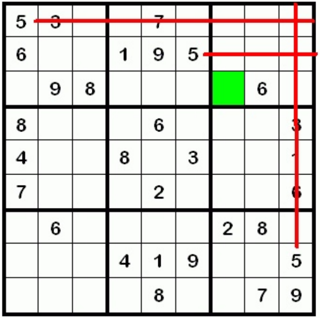
Sudoku

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

c wikipedia.com

```
sudoko(Matrix) <=>
  gendomain(Matrix,1..9),
  Matrix=[A1,A2,...,A9,B1,...,I9],
  alldiff(A1,A2,A3,A4,A5,A6,A7,A8,A9),
  ...,
  alldiff(A1,B1,C1,D1,E1,F1,G1,H1,I1),
  ...,
  alldiff(A1,B1,C1,A2,B2,C2,A3,B3,C3),
  ...,
  labeling(Matrix).
```

Sudoku



c wikipedia.com

```
sudoko(Matrix) <=>
  gendomain(Matrix,1..9),
  Matrix=[A1,A2,...,A9,B1,...,I9],
  alldiff(A1,A2,A3,A4,A5,A6,A7,A8,A9),
  ...,
  alldiff(A1,B1,C1,D1,E1,F1,G1,H1,I1),
  ...,
  alldiff(A1,B1,C1,A2,B2,C2,A3,B3,C3),
  ...,
  labeling(Matrix).
```

Mortgage

D: Amount of Loan, Debt, Principal
T: Duration of loan in months
I: Interest rate per month
R: Rate of payments per month
S: Balance of debt after T months

```
mortgage(D, T, I, R, S) <=>
       T = 0,
       D = S
         ;
       T > 0,
       T1 = T - 1,
       D1 = D + D*I - R,
       mortgage(D1, T1, I, R, S).
```

Mortgage II

```
mortgage(D, T, I, R, S) <=>
       T = 0, D = S
         ;
       T > 0, T1 = T - 1, D1 = D + D*I - R,
       mortgage(D1, T1, I, R, S).
```

- `mortgage(100000,360,0.01,1025,S)` yields `S=12625.90`.
- `mortgage(D,360,0.01,1025,0)` yields `D=99648.79`.
- `mortgage(100000,T,0.01,1025,S), S=<0` yields `T=374, S=-807.96`.
- `mortgage(D,360,0.01,R,0)` yields `R=0.0102861198*D`.

Advantages of Constraint Logic Programming

Theoretical
Logical Foundation – First-Order Logic

Conceptual
Sound Modeling

Practical
Efficient Algorithms/Implementations
Combination of different Solvers

Early Commercial Applications (in the 90s)

- Lufthansa: Short-term staff planning.
- Hongkong Container Harbor: Resource planning.
- Renault: Short-term production planning.
- Nokia: Software configuration for mobile phones.
- Airbus: Cabin layout.
- Siemens: Circuit verification.
- Caisse d'epargne: Portfolio management.

In Decision Support Systems for **Planning and Configuration**, for **Design and Analysis**.

Part II

CHR Constraint Handling Rules

Example Partial Order Constraint

$$
\begin{array}{rcll}
X \leq X & \Leftrightarrow & true & \text{(reflexivity)} \\
X \leq Y \wedge Y \leq X & \Leftrightarrow & X = Y & \text{(antisymmetry)} \\
X \leq Y \wedge Y \leq Z & \Rightarrow & X \leq Z & \text{(transitivity)}
\end{array}
$$

$$
\begin{array}{cl}
\underline{A \leq B} \wedge \underline{B \leq C} \wedge C \leq A & \\
\downarrow & \text{(transitivity)} \\
A \leq B \wedge B \leq C \wedge \underline{C \leq A} \wedge \underline{A \leq C} & \\
\downarrow & \text{(antisymmetry)} \\
A \leq B \wedge B \leq C \wedge \underline{A = C} & \\
\downarrow & \text{(built-in solver)} \\
\underline{A \leq B} \wedge \underline{B \leq A} \wedge A = C & \\
\downarrow & \text{(antisymmetry)} \\
A = B \wedge A = C &
\end{array}
$$

Example Partial Order Constraint

$$
\begin{array}{rcll}
X \leq X & \Leftrightarrow & true & \text{(reflexivity)} \\
X \leq Y \wedge Y \leq X & \Leftrightarrow & X = Y & \text{(antisymmetry)} \\
X \leq Y \wedge Y \leq Z & \Rightarrow & X \leq Z & \text{(transitivity)}
\end{array}
$$

$$
\begin{array}{cl}
\underline{A \leq B} \wedge \underline{B \leq C} \wedge C \leq A & \\
\downarrow & \text{(transitivity)} \\
A \leq B \wedge B \leq C \wedge \underline{C \leq A} \wedge \underline{A \leq C} & \\
\downarrow & \text{(antisymmetry)} \\
A \leq B \wedge B \leq C \wedge \underline{A = C} & \\
\downarrow & \text{(built-in solver)} \\
\underline{A \leq B} \wedge \underline{B \leq A} \wedge A = C & \\
\downarrow & \text{(antisymmetry)} \\
A = B \wedge A = C &
\end{array}
$$

Example Partial Order Constraint

$$X \leq X \Leftrightarrow true \quad \text{(reflexivity)}$$
$$X \leq Y \wedge Y \leq X \Leftrightarrow X = Y \quad \text{(antisymmetry)}$$
$$X \leq Y \wedge Y \leq Z \Rightarrow X \leq Z \quad \text{(transitivity)}$$

$$\underline{A \leq B} \wedge \underline{B \leq C} \wedge C \leq A$$
$$\downarrow \quad \text{(transitivity)}$$
$$A \leq B \wedge B \leq C \wedge \underline{C \leq A} \wedge \underline{A \leq C}$$
$$\downarrow \quad \text{(antisymmetry)}$$
$$A \leq B \wedge B \leq C \wedge \underline{A = C}$$
$$\downarrow \quad \text{(built-in solver)}$$
$$\underline{A \leq B} \wedge \underline{B \leq A} \wedge A = C$$
$$\downarrow \quad \text{(antisymmetry)}$$
$$A = B \wedge A = C$$

Example Partial Order Constraint

$$
\begin{array}{rcll}
X \leq X & \Leftrightarrow & true & \text{(reflexivity)} \\
X \leq Y \wedge Y \leq X & \Leftrightarrow & X = Y & \text{(antisymmetry)} \\
X \leq Y \wedge Y \leq Z & \Rightarrow & X \leq Z & \text{(transitivity)}
\end{array}
$$

$$
\begin{array}{cl}
\underline{A \leq B} \wedge \underline{B \leq C} \wedge C \leq A & \\
\downarrow & \text{(transitivity)} \\
A \leq B \wedge B \leq C \wedge \underline{C \leq A} \wedge \underline{A \leq C} & \\
\downarrow & \text{(antisymmetry)} \\
A \leq B \wedge B \leq C \wedge \underline{A = C} & \\
\downarrow & \text{(built-in solver)} \\
\underline{A \leq B} \wedge \underline{B \leq A} \wedge A = C & \\
\downarrow & \text{(antisymmetry)} \\
A = B \wedge A = C &
\end{array}
$$

Example Partial Order Constraint

$$X \leq X \quad \Leftrightarrow \quad true \qquad \text{(reflexivity)}$$
$$X \leq Y \wedge Y \leq X \quad \Leftrightarrow \quad X = Y \qquad \text{(antisymmetry)}$$
$$X \leq Y \wedge Y \leq Z \quad \Rightarrow \quad X \leq Z \qquad \text{(transitivity)}$$

$$\underline{A \leq B} \wedge \underline{B \leq C} \wedge C \leq A$$
$$\downarrow \qquad \text{(transitivity)}$$
$$A \leq B \wedge B \leq C \wedge \underline{C \leq A} \wedge \underline{A \leq C}$$
$$\downarrow \qquad \text{(antisymmetry)}$$
$$A \leq B \wedge B \leq C \wedge \underline{A = C}$$
$$\downarrow \qquad \text{(built-in solver)}$$
$$\underline{A \leq B} \wedge \underline{B \leq A} \wedge A = C$$
$$\downarrow \qquad \text{(antisymmetry)}$$
$$A = B \wedge A = C$$

Example Partial Order Constraint

$$
\begin{array}{rcll}
X \leq X & \Leftrightarrow & true & \text{(reflexivity)} \\
X \leq Y \wedge Y \leq X & \Leftrightarrow & X = Y & \text{(antisymmetry)} \\
X \leq Y \wedge Y \leq Z & \Rightarrow & X \leq Z & \text{(transitivity)}
\end{array}
$$

$$
\begin{array}{cl}
\underline{A \leq B} \wedge \underline{B \leq C} \wedge C \leq A & \\
\downarrow & \text{(transitivity)} \\
A \leq B \wedge B \leq C \wedge \underline{C \leq A} \wedge \underline{A \leq C} & \\
\downarrow & \text{(antisymmetry)} \\
A \leq B \wedge B \leq C \wedge \underline{A = C} & \\
\downarrow & \text{(built-in solver)} \\
\underline{A \leq B} \wedge \underline{B \leq A} \wedge A = C & \\
\downarrow & \text{(antisymmetry)} \\
A = B \wedge A = C &
\end{array}
$$

Operational Semantics

Apply rules until exhaustion in any order (fixpoint computation).

Simplify

If	$(H \Leftrightarrow C \mid B)$ rule with renamed fresh variables $\bar{x}$
and	$CT \models G_{builtin} \rightarrow \exists \bar{x}(H{=}H' \wedge C)$
then	$H' \wedge G \mapsto G \wedge H{=}H' \wedge B$

Propagate

If	$(H \Rightarrow C \mid B)$ rule with renamed fresh variables $\bar{x}$
and	$CT \models G_{builtin} \rightarrow \exists \bar{x}(H{=}H' \wedge C)$
then	$H' \wedge G \mapsto H' \wedge G \wedge H{=}H' \wedge B$

Operational Semantics

Apply rules until exhaustion in any order (fixpoint computation).

Simplify

If $(H \Leftrightarrow C \mid B)$ rule with renamed fresh variables $\bar{x}$
and $CT \models G_{builtin} \rightarrow \exists \bar{x}(H{=}H' \wedge C)$
then $H' \wedge G \mapsto G \wedge H{=}H' \wedge B$

Propagate

If $(H \Rightarrow C \mid B)$ rule with renamed fresh variables $\bar{x}$
and $CT \models G_{builtin} \rightarrow \exists \bar{x}(H{=}H' \wedge C)$
then $H' \wedge G \mapsto H' \wedge G \wedge H{=}H' \wedge B$

Operational Semantics

Apply rules until exhaustion in any order (fixpoint computation).

Simplify

If	$(H \Leftrightarrow C \mid B)$ rule with renamed fresh variables $\bar{x}$
and	$CT \models G_{builtin} \rightarrow \exists \bar{x}(H{=}H' \wedge C)$
then	$H' \wedge G \mapsto G \wedge H{=}H' \wedge B$

Propagate

If	$(H \Rightarrow C \mid B)$ rule with renamed fresh variables $\bar{x}$
and	$CT \models G_{builtin} \rightarrow \exists \bar{x}(H{=}H' \wedge C)$
then	$H' \wedge G \mapsto H' \wedge G \wedge H{=}H' \wedge B$

Anytime Algorithm

Computation can be interrupted and restarted at any time.
Intermediate results approximate final result.

$$\underline{A{\leq}B} \wedge \underline{B{\leq}C} \wedge C{\leq}A$$

$$\downarrow \quad \text{(transitivity)}$$

$$A{\leq}B \wedge B{\leq}C \wedge \underline{C{\leq}A} \wedge \underline{A{\leq}C}$$

$$\downarrow \quad \text{(antisymmetry)}$$

$$A{\leq}B \wedge B{\leq}C \wedge \underline{A{=}C}$$

$$\downarrow \quad \text{(built-in solver)}$$

$$\underline{A{\leq}B} \wedge \underline{B{\leq}A} \wedge A{=}C$$

$$\downarrow \quad \text{(antisymmetry)}$$

$$A{=}B \wedge A{=}C$$

Anytime Algorithm

Computation can be interrupted and restarted at any time.
Intermediate results approximate final result.

$$\underline{A \leq B} \wedge \underline{B \leq C} \wedge C \leq A$$
$$\downarrow \qquad \text{(transitivity)}$$
$$A \leq B \wedge B \leq C \wedge \underline{C \leq A} \wedge \underline{A \leq C}$$
$$\downarrow \qquad \text{(antisymmetry)}$$
$$A \leq B \wedge B \leq C \wedge \underline{A = C}$$
$$\downarrow \qquad \text{(built-in solver)}$$
$$\underline{A \leq B} \wedge \underline{B \leq A} \wedge A = C$$
$$\downarrow \qquad \text{(antisymmetry)}$$
$$A = B \wedge A = C$$

Anytime Algorithm

Computation can be interrupted and restarted at any time.
Intermediate results approximate final result.

$$\underline{A \leq B} \wedge \underline{B \leq C} \wedge C \leq A$$

$$\downarrow \qquad \text{(transitivity)}$$

$$A \leq B \wedge B \leq C \wedge \underline{C \leq A} \wedge \underline{A \leq C}$$

$$\downarrow \qquad \text{(antisymmetry)}$$

$$A \leq B \wedge B \leq C \wedge \underline{A = C}$$

$$\downarrow \qquad \text{(built-in solver)}$$

$$\underline{A \leq B} \wedge \underline{B \leq A} \wedge A = C$$

$$\downarrow \qquad \text{(antisymmetry)}$$

$$A = B \wedge A = C$$

Online Algorithm

The complete input is initially unknown.
The input data arrives incrementally during computation.
No recomputation from scratch necessary.

Monotonicity and Incrementality

If	G	$\longmapsto$	G'
then	$G \wedge C$	$\longmapsto$	$G' \wedge C$

$$\underline{A\leq B} \wedge \underline{B\leq C} \wedge C\leq A$$
$$\downarrow \qquad \text{(transitivity)}$$
$$A\leq B \wedge B\leq C \wedge \underline{A\leq C} \wedge \underline{C\leq A}$$
$$\downarrow \qquad \text{(antisymmetry)}$$
$$A\leq B \wedge B\leq C \wedge \underline{A=C}$$
$$\downarrow$$
$$\ldots$$

Online Algorithm

The complete input is initially unknown.
The input data arrives incrementally during computation.
No recomputation from scratch necessary.

Monotonicity and Incrementality

If	G	$\longmapsto$	G'
then	$G \wedge C$	$\longmapsto$	$G' \wedge C$

$$\underline{A \leq B} \wedge \underline{B \leq C} \wedge C \leq A$$

$$\downarrow \qquad \text{(transitivity)}$$

$$A \leq B \wedge B \leq C \wedge \underline{A \leq C} \wedge \underline{C \leq A}$$

$$\downarrow \qquad \text{(antisymmetry)}$$

$$A \leq B \wedge B \leq C \wedge \underline{A = C}$$

$$\downarrow$$

$$\ldots$$

Online Algorithm

The complete input is initially unknown.
The input data arrives incrementally during computation.
No recomputation from scratch necessary.

Monotonicity and Incrementality

If	G	$\longmapsto$	G'
then	$G \wedge C$	$\longmapsto$	$G' \wedge C$

$$\underline{A\leq B} \wedge \underline{B\leq C} \wedge C\leq A$$
$$\downarrow \qquad \text{(transitivity)}$$
$$A\leq B \wedge B\leq C \wedge \underline{A\leq C} \wedge \underline{C\leq A}$$
$$\downarrow \qquad \text{(antisymmetry)}$$
$$A\leq B \wedge B\leq C \wedge \underline{A=C}$$
$$\downarrow$$
$$\ldots$$

Online Algorithm

The complete input is initially unknown.
The input data arrives incrementally during computation.
No recomputation from scratch necessary.

Monotonicity and Incrementality

$$\begin{array}{llcl} \text{If} & G & \longmapsto & G' \\ \text{then} & G \wedge C & \longmapsto & G' \wedge C \end{array}$$

$$\begin{array}{c} \underline{A{\leq}B} \wedge \underline{B{\leq}C} \wedge C{\leq}A \\ \downarrow \\ A{\leq}B \wedge B{\leq}C \wedge \underline{A{\leq}C} \wedge \underline{C{\leq}A} \\ \downarrow \\ A{\leq}B \wedge B{\leq}C \wedge \underline{A{=}C} \\ \downarrow \\ \ldots \end{array}$$

(transitivity)

(antisymmetry)

Concurrency

Rules can be applied in parallel to different parts of the problem.

$$
\begin{array}{llcl}
\text{If} & A & \longmapsto & B \\
\text{and} & C & \longmapsto & D \\
\text{then} & A \wedge C & \longmapsto & B \wedge D
\end{array}
$$

$$
\begin{array}{ccc}
\underline{A \leq B} \wedge \underline{B \leq C} & \wedge & C \leq D \wedge D \leq A \\
\downarrow & & \downarrow \\
A \leq B \wedge B \leq C \wedge \underline{A \leq C} & \wedge & C \leq D \wedge D \leq A \wedge \underline{C \leq A} \\
& \downarrow & \\
& \ldots \; A = C \; \ldots &
\end{array}
$$

Concurrency

Rules can be applied in parallel to different parts of the problem.

If	A	$\longmapsto$	B
and	C	$\longmapsto$	D
then	$A \wedge C$	$\longmapsto$	$B \wedge D$

$$\underline{A{\leq}B} \wedge \underline{B{\leq}C} \qquad \wedge \qquad C{\leq}D \wedge D{\leq}A$$

$$\downarrow \qquad\qquad\qquad\qquad \downarrow$$

$$A{\leq}B \wedge B{\leq}C \wedge \underline{A{\leq}C} \qquad \wedge \qquad C{\leq}D \wedge D{\leq}A \wedge \underline{C{\leq}A}$$

$$\downarrow$$

$$\ldots \; A{=}C \; \ldots$$

Concurrency

Rules can be applied in parallel to different parts of the problem.

$$\begin{array}{llcl} \text{If} & A & \longmapsto & B \\ \text{and} & C & \longmapsto & D \\ \text{then} & A \wedge C & \longmapsto & B \wedge D \end{array}$$

$$\begin{array}{ccc} \underline{A{\leq}B} \wedge \underline{B{\leq}C} & \wedge & C{\leq}D \wedge D{\leq}A \\ \downarrow & & \downarrow \\ A{\leq}B \wedge B{\leq}C \wedge \underline{A{\leq}C} & \wedge & C{\leq}D \wedge D{\leq}A \wedge \underline{C{\leq}A} \\ & \downarrow & \\ & \ldots\ A{=}C\ \ldots & \end{array}$$

Concurrency

Rules can be applied in parallel to different parts of the problem.

$$
\begin{array}{llll}
\text{If} & A & \longmapsto & B \\
\text{and} & C & \longmapsto & D \\
\text{then} & A \wedge C & \longmapsto & B \wedge D
\end{array}
$$

$$
\begin{array}{ccc}
\underline{A\leq B} \wedge \underline{B\leq C} & \wedge & C\leq D \wedge D\leq A \\
\downarrow & & \downarrow \\
A\leq B \wedge B\leq C \wedge \underline{A\leq C} & \wedge & C\leq D \wedge D\leq A \wedge \underline{C\leq A} \\
& \downarrow & \\
& \ldots\ A{=}C\ \ldots &
\end{array}
$$

Linear Polynomial Equations

Equations of the form $a_1 x_1 + \ldots + a_n x_n + b = 0$.
Solved form: leftmost variable occurs only once.
Reach solved normal form by **variable elimination**.

```
A1*X+P1=0 ∧ XP=0  ⇔
    find(A2*X,XP,P2) |
    compute(P2-(P1/A1)*A2,P3) ∧
    A1*X+P1=0 ∧ P3=0.

B=0  ⇔  number(B) |  zero(B).
```

```
1*X+3*Y+5=0 ∧  3*X+2*Y+8=0
 compute((2*Y+8) - ((3*Y+5)/1)*3,P3)   % P3=-7*Y+ -7
1*X+3*Y+5=0 ∧  -7*Y+ -7=0  % Y=-1
 compute((1*X+5) - ((-7)/-7)*3,P3')    % P3'=1*X+2
1*X+2=0 ∧       -7*Y+ -7=0  % X=-2
```

Linear Polynomial Equations

Equations of the form $a_1x_1 + \ldots + a_nx_n + b = 0$.
Solved form: leftmost variable occurs only once.
Reach solved normal form by **variable elimination**.

```
A1*X+P1=0 ∧ XP=0  ⇔
   find(A2*X,XP,P2) |
   compute(P2-(P1/A1)*A2,P3) ∧
   A1*X+P1=0 ∧ P3=0.

B=0  ⇔  number(B) |  zero(B).

         1*X+3*Y+5=0 ∧  3*X+2*Y+8=0
          compute((2*Y+8) - ((3*Y+5)/1)*3,P3)  % P3=-7*Y+ -7
         1*X+3*Y+5=0 ∧  -7*Y+ -7=0  % Y=-1
          compute((1*X+5) - ((-7)/-7)*3,P3')   % P3'=1*X+2
         1*X+2=0 ∧      -7*Y+ -7=0  % X=-2
```

Linear Polynomial Equations

Equations of the form $a_1 x_1 + \ldots + a_n x_n + b = 0$.
Solved form: leftmost variable occurs only once.
Reach solved normal form by **variable elimination**.

```
A1*X+P1=0 ∧ XP=0  ⇔
   find(A2*X,XP,P2) |
   compute(P2-(P1/A1)*A2,P3) ∧
   A1*X+P1=0 ∧ P3=0.
```

```
B=0  ⇔  number(B) |  zero(B).
```

```
1*X+3*Y+5=0 ∧  3*X+2*Y+8=0
 compute((2*Y+8) - ((3*Y+5)/1)*3,P3)   % P3=-7*Y+ -7
1*X+3*Y+5=0 ∧  -7*Y+ -7=0  % Y=-1
 compute((1*X+5) - ((-7)/-7)*3,P3')    % P3'=1*X+2
1*X+2=0 ∧      -7*Y+ -7=0  % X=-2
```

Linear Polynomial Equations

Equations of the form $a_1x_1 + \ldots + a_nx_n + b = 0$.
Solved form: leftmost variable occurs only once.
Reach solved normal form by **variable elimination**.

```
A1*X+P1=0 ∧ XP=0  ⇔
   find(A2*X,XP,P2) |
   compute(P2-(P1/A1)*A2,P3) ∧
   A1*X+P1=0 ∧ P3=0.
```

```
B=0  ⇔  number(B) |  zero(B).
```

```
1*X+3*Y+5=0 ∧  3*X+2*Y+8=0
 compute((2*Y+8) - ((3*Y+5)/1)*3,P3)  % P3=-7*Y+ -7
1*X+3*Y+5=0 ∧  -7*Y+ -7=0  % Y=-1
 compute((1*X+5) - ((-7)/-7)*3,P3')   % P3'=1*X+2
1*X+2=0 ∧      -7*Y+ -7=0  % X=-2
```

Confluence

Given a goal, every computation leads to the same result no matter what rules are applied.
A decidable, sufficient and necessary condition for confluence of terminating CHR programs through joinability of critical pairs.

$$
\begin{array}{rcll}
X \leq X & \Leftrightarrow & \mathit{true} & \text{(reflexivity)} \\
X \leq Y \wedge Y \leq X & \Leftrightarrow & X = Y & \text{(antisymmetry)}
\end{array}
$$

Start from overlapping minimal states

$$
\begin{tikzcd}
 & A \leq A \wedge A \leq A \arrow[dl, "\text{reflexivity}"'] \arrow[dr, "\text{antisymmetry}"] & \\
A \leq A \arrow[dr, "\text{reflexivity}"'] & & A = A \arrow[dl, "\text{built-in}"] \\
 & \mathit{true} &
\end{tikzcd}
$$

Confluence

Given a goal, every computation leads to the same result no matter what rules are applied.
A decidable, sufficient and necessary condition for confluence of terminating CHR programs through joinability of critical pairs.

$$\begin{array}{rcll} X \leq X & \Leftrightarrow & \mathit{true} & \text{(reflexivity)} \\ X \leq Y \wedge Y \leq X & \Leftrightarrow & X = Y & \text{(antisymmetry)} \end{array}$$

Start from overlapping minimal states

$$\begin{tikzcd} & A \leq A \wedge A \leq A \arrow[dl, "\text{reflexivity}"'] \arrow[dr, "\text{antisymmetry}"] & \\ A \leq A \arrow[dr, "\text{reflexivity}"'] & & A = A \arrow[dl, "\text{built-in}"] \\ & \mathit{true} & \end{tikzcd}$$

Confluence

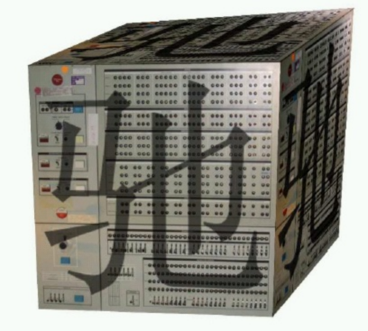
Given a goal, every computation leads to the same result no matter what rules are applied.
A decidable, sufficient and necessary condition for confluence of terminating CHR programs through joinability of critical pairs.

$$
\begin{array}{rcll}
X \leq X & \Leftrightarrow & \mathit{true} & \text{(reflexivity)} \\
X \leq Y \wedge Y \leq X & \Leftrightarrow & X = Y & \text{(antisymmetry)}
\end{array}
$$

Start from overlapping minimal states

$$
\begin{tikzcd}
 & A \leq A \wedge A \leq A \arrow[dl, "\text{reflexivity}"'] \arrow[dr, "\text{antisymmetry}"] & \\
A \leq A \arrow[dr, "\text{reflexivity}"'] & & A = A \arrow[dl, "\text{built-in}"] \\
 & \mathit{true} &
\end{tikzcd}
$$

CHR Machine



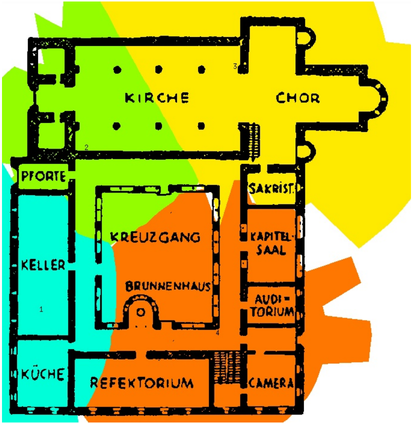
c Jon Sneyers, K.U. Leuven

Analogous to RAM and Turing machines.
CHR is turing-complete.
Every algorithm can be implemented in CHR with best known time and space complexity.
[Sneyers,Schrijvers,Demoen, CHR'05]

Part III

Applications

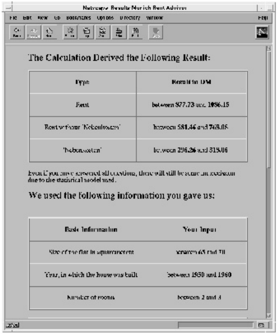
POPULAR - Planning Cordless Communication



T. Frühwirth, P. Brisset
Optimal Placement of Base Stations in Wireless Indoor Communication Networks, IEEE Intelligent Systems Magazine 15(1), 2000.

Voted Among Most Innovative Telecom Applications of the Year by IEEE Expert Magazine, Winner of CP98 Telecom Application Award.

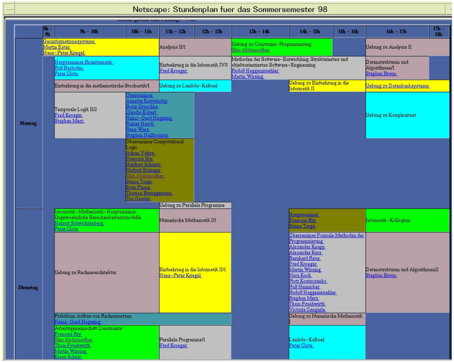
MRA - The Munich Rent Advisor



T. Frühwirth,
S. Abdennadher
The Munich Rent Advisor,
Journal of Theory and Practice of Logic Programming, 2000.

Most Popular Constraint-Based Internet Application.

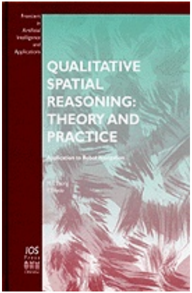
University Course Timetabling



S. Abdennadher, M. Saft, S. Will
Classroom Assignment using Constraint Logic Programming,
PACLP 2000.

Operational at University of Munich. Room-Allocation for 1000 Lectures a Week.

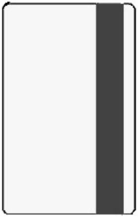
Spatio-Temporal Reasoning



M. T. Escrig, F. Toledo,
Universidad Jaume I, Castellun, Spain.
Qualitative Spatial Reasoning: Theory and Practice, Application to Robot Navigation, IOS Press, 1998.
Qualitative Spatial Reasoning on 3D Orientation Point Objects, QR2002.
Integrates orientation, distance, cardinal directions over points as well as extended objects.

- Spatio-Temporal Annotated CLP - A. Raffaeta, Univ. Venice.
- Diagrammatic Reasoning - B. Meyer, Monash Melbourne.
- RCC Reasoning - B. Bennet, A.G. Cohn, Leeds UK.
- PMON logic for dynamical temporal systems - E. Sandewall, Linkoeping Univ.
- GRF Temporal Reasoning - G. Dondossola, E. Ratto, CISE Milano.

Testing and Verification



Model Based Testing for Real:
The Inhouse Card Case Study,
A. Pretschner, O. Slotosch, E. Aiglstorfer, S. Kriebel,
TU Munich,
Journal on Software Tools for Technology Transfer
(STTT) 5:2-3, Springer 2004.

- Automatic Generation of Test Data - J. Harm, University Rostock, Germany.
- Executable Z-Specifications - P. Stuckey, Ph. Dart, University Melbourne.

Agents and Actions

FLUX: A Logic Programming Method for Reasoning Agents,
Michael Thielscher, TPLP CHR Special Issue 2005.
Fluent Calculus, Reasoning about Actions, Robotics.

Specification and Verification of Agent Interaction...
Alberti, Chesani, Gavanelli, Lamma, Mello, Torroni, ACM SAC 2004.
Social integrity constraints on agent behaviour.

- Multi Agent Systems Using Constrains Handling Rules, IC-AI 2002 - B. Bauer, M. Berger, Siemens Munich, Germany - S. Hainzer, Uni Linz, Austria.
- PMON logic for dynamical temporal systems with actions and change - M. Bjgareland, E. Sandewall, Linkoeping University, Sweden.

Semantic Web

The COntext INterchange Project

COIN Context Interchange Project,
Stuart E. Madnick, MIT Cambridge.
Reasoning About Temporal Context Using Ontology and Abductive CLP,
PPSWR 2004 LNCS 3208.

Semantic Web Reasoning for Ontology-Based Integration of Resources,
Liviu Badea, Doina Tilivea and Anca Hotaran, PPSWR 2004 LNCS 3208.

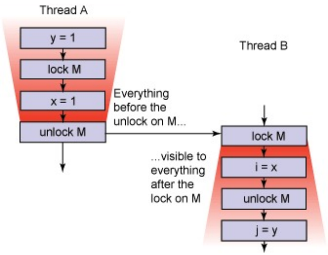
- S. Bressan, C.H. Goh, S. Madnick, M. Siegel et. al.

Context Knowledge Representation and Reasoning in the Context Interchange System, Applied Intelligence, Vol 13:2, 2000;

Context Interchange...for the intelligent integration of information, ACM Transactions on Information Systems, 1999.

Java Memory Machine

JMM by Vijay Saraswat, IBM TJ Watson Research and Penn State Univ.
Implementation JMMSolve by Tom Schrijvers, K.U. Leuven, Belgium



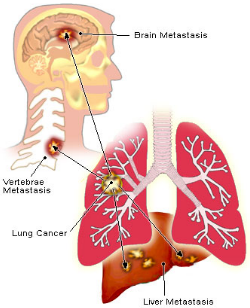
Conditional Read

```
Xr = (Cond)?Xw1:Xi

ite(true,Xr,Xw1,Xi)  <=> Xr = Xw1.
ite(false,Xr,Xw1,Xi) <=> Xr = Xi.
ite(Cond,Xr,X,X)     <=> Xr = X.
```

Lung Cancer Diagnosis

Veronica Dahl, Simon Fraser University, Vancouver, Canada.
Lung cancer is leading cause of cancer death, very low survival rate.
Use bio-markers indicating gene mutations to diagnose lung cancer.



Lung Cancer and Metastasis

Concept Formation Rules (CFR) in CHR.
Retractable constraints.

```
age(X,A),history(X,smoker),
serum_data(X,marker_type) <=>
marker(X,marker_type,P,B),
probability(P,X,B) |
possible_lung_cancer(yes,X).
```

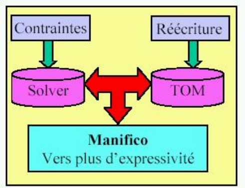
Multimedia Transformation Engine for Web Presentations

Joost Geurts, University of Amsterdam.
Automatic generation of interactive, time-based and media centric WWW presentations from semi-structured multimedia databases.

Business Rules for Optimization

MANIFICO - Francois Fages, Claude Kirchner, Hassan Ait-Kaci,...France

Business Rule: defines or constrains behavior or structure of business.
"A car must be available to be assigned to a rental agreement".

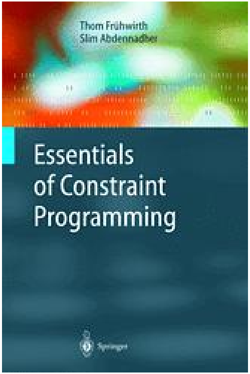


DERBY EU Car Rent Case in CHR, O. Bouissou.

```
reservation(Renter,Group,From,To),
available(car(Id,Group,...),From) <=>...
        rentagreement(Renter,Id,From,To).
```

References

Google "constraint handling rules"



Essentials of Constraint Programming
Thom Frühwirth,
Slim Abdennadher
Springer, 2003.

Constraint-Programmierung
Lehrbuch
Thom Frühwirth,
Slim Abdennadher
Springer, 1997.