Constraint Handling Rules
What's Next?

Prof. Dr. Thom Frühwirth | CHR Working Week, October 2009 | University of Ulm, Germany

Table of Contents

Renaissance of rule-based approaches

Results on rule-based system re-used and re-examined for

- Business rules and Workflow systems
- Semantic Web (e.g. validating forms, ontology reasoning, OWL)
- UML (e.g. OCL invariants) and extensions (e.g. ATL)
- Computational Biology (e.g. protein folding, genome analysis)
- Medical Diagnosis and Decision Support
- Software Verification and Security (e.g. access policies)
- Solving Design and Configuration Problems

The success of Constraint Handling Rules

CHR - an essential unifying computational formalism?

- CHR is a logic *and* a programming language
- CHR can express any algorithm with optimal complexity
- CHR supports reasoning and program analysis
- CHR programs are anytime, online and concurrent algorithms
- CHR programs are efficient and very fast
- CHR has numerous applications from academia to industry

$\Rightarrow$ **CHR - a Lingua Franca for computer science:**
*** The first formalism and the first language for CS students**
*** Reasoning formalism and programming language for research**

The Magic Triangle

- Basic Research
- Training and Education (not discussed here)
- Applications and Demonstrators

CHR Research at University Ulm

Basic research themes

- Linear Logic Semantics
- (Massively) parallel CHR
- Embeddings of Formalisms and Languages

Make classic results more accessable

Research Topics (Frank Raiser)

Linear Logic and CHR

- joint work with Hariolf Betz
- goes hand-in-hand with simplifying operational semantics
- we aim for closer resemblance of LL and CHR
- also related to work on persistent constraints

Research Topics (Frank Raiser)

Massively Parallel CHR

- still in very early stage
- assumption: CHR on multi-core CPUs and number of cores skyrockets in future
- persistent constraints inherently suitable for massively concurrent execution (no conflicts possible)
- $\Rightarrow$ *Sorting in constant time*

Research Topics (Frank Raiser)

Program Analysis

- focus on confluence and program equivalence
- simplification of operational semantics
  - facilitates program analysis
  - *Simplicity is prerequisite for reliability.* (E.W.Dijkstra)

Research Topics (Frank Raiser)

Dynamic Programming

- CHR allows straightforward translation of DP recursion formulae into rules
- Problem up to now: runtime
  - complicated control mechanisms required
  - makes heavy use of indexing
- CHR^{rp} seems a suitable solution
- *almost* optimal runtime (and space) complexity (i.e., there is some work left to do)

Research Topics (Frank Raiser)

Talk – State Equivalence and Persistent Constraints

- Covers three of the previous research areas:
  - simplified operational semantics
  - better declarativity, closer to linear logic
  - persistent constraints perfect for massive parallelism

Description Logic with Rules in CHR (Fruehwirth)

Straightforward integration of DL, rules and constraints.
DL in CHR: shorter than formal specification!
Correct, confluent, concurrent, anytime, online algorithm.

`and`:	**If**	$x : C_1 \sqcap C_2 \in \mathcal{A}$ and $\{x : C_1, x : C_2\} \not\subseteq \mathcal{A}$
	then	$\mathcal{A} \rightarrow_{\sqcap} \mathcal{A} \cup \{x : C_1, x : C_2\}$
`or`:	**If**	$x : C_1 \sqcup C_2 \in \mathcal{A}$ and $\{x : C_1, x : C_2\} \cap \mathcal{A} = \emptyset$
	then	$\mathcal{A} \rightarrow_{\sqcup} \mathcal{A} \cup \{x : D\}$ for some $D \in \{C_1, C_2\}$
`some`:	**If**	$x : \exists R.D \in \mathcal{A}$ and there is no y with $\{(x, y) : R, y : D\} \subseteq \mathcal{A}$
	then	$\mathcal{A} \rightarrow_{\exists} \mathcal{A} \cup \{(x, y) : R,\ y : D\}$ for a fresh individual y
`all`:	**If**	$x : \forall R.D \in \mathcal{A}$ and there is a y with $(x, y) : R \in \mathcal{A}$ and $y : D \notin \mathcal{A}$
	then	$\mathcal{A} \rightarrow_{\forall} \mathcal{A} \cup \{y : D\}$

Figure: The completion rules for $\mathcal{ALC}$

Description Logic with Rules in CHR (Fruehwirth)

Straightforward integration of DL, rules and constraints.
DL in CHR: shorter than formal specification!
Correct, confluent, concurrent, anytime, online algorithm.

```
and   @ I:S1 and S2 <=> I:S1, I:S2
or    @ I:S1 or S2 <=> (I:S1 ; I:S2)
some  @ I:some R is S <=> (I,J):R, J:S
all   @ I:all R is S, (I,J):R ==> J:S
```

Figure: CHR Rules for $\mathcal{ALC}$

Major CHR Research Groups

- K.U. Leuven, Beligum; Jon Sneyers, Peter van Weert
- University of Ulm, Germany; Thom Fruehwirth
- U. of Melbourne, NICTA Victoria Lab., Australia; Peter Stuckey
- National U., Singapore; Edmund Lam
- Roskilde U., DIKU, Denmark; Henning Christiansen
- Simon Fraser U. Vancouver, Canada; Veronica Dahl
- INRIA Paris-Rocquencourt, France; Francois Fages
- GUC Cairo, Egypt; Slim Abdennadher
- U. of Ferrara; Evelina Lamma
- U. of Bologna; Maurizio Gabbrielli
- U. Federal de Pernambuco, Recife, Brazil; Jacques Robin
- Fraunhofer FIRST, Berlin; Armin Wolf

CHR and Tangible User Interfaces

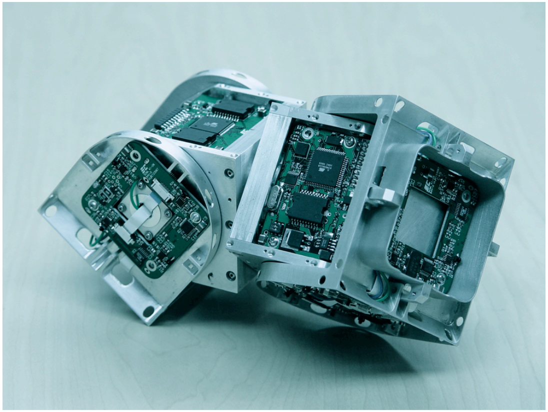
- **Reac-tables**
  - Typotisch, Uni Ulm (*)
  - TTable, Uni Ulm (*)
- **Robots**
  - Small E-Puck Robots, Mohammed Oubbati, University Ulm
  - Superbots, Mike Rubenstein, University of Southern California, USA
  - Robotic Sailing, Roland Stelzer, Innoc, Austria (*)

(*) Talk during CHR Working Week, Oct 09, Ulm

E-Puck Robots at Uni Ulm

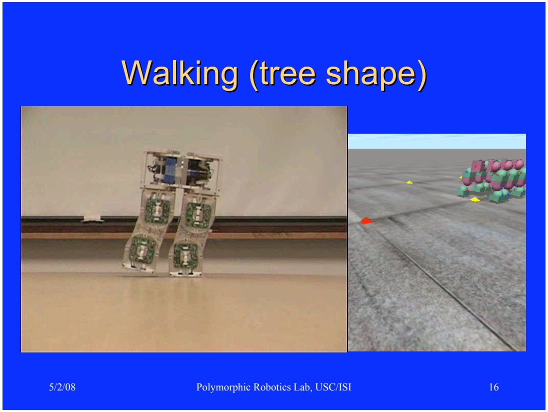
Superbots - Self-Reconfigurable Robots

Wei-Min Shen, Mike Rubenstein, Univ. of Southern California, USA



Superbots - Self-Reconfigurable Robots

Wei-Min Shen, Mike Rubenstein, Univ. of Southern California, USA



Walking (tree shape)

5/2/08 Polymorphic Robotics Lab, USC/ISI 16

Superbots - Self-Reconfigurable Robots

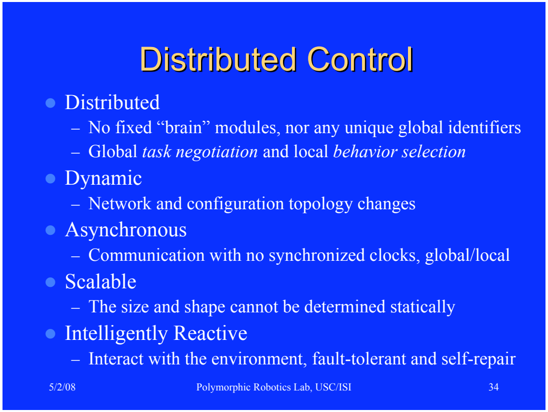
Wei-Min Shen, Mike Rubenstein, Univ. of Southern California, USA

Climb Sand Dune Experiments

5/2/08 Polymorphic Robotics Lab, USC/ISI 27

Superbots - Self-Reconfigurable Robots

Wei-Min Shen, Mike Rubenstein, Univ. of Southern California, USA



Distributed Control

- Distributed
  - No fixed “brain” modules, nor any unique global identifiers
  - Global *task negotiation* and local *behavior selection*
- Dynamic
  - Network and configuration topology changes
- Asynchronous
  - Communication with no synchronized clocks, global/local
- Scalable
  - The size and shape cannot be determined statically
- Intelligently Reactive
  - Interact with the environment, fault-tolerant and self-repair

5/2/08 Polymorphic Robotics Lab, USC/ISI 34

Current CHR Implementations

CHR in Logic Programming

- K.U. Leuven CHR for Prolog, Schrijvers
  - SWI, Wielemaker; Sictus, Carlson; YAP; hProlog, Demoen; XSB, Warren; B-Prolog, Zhou; Ciao, Hermenegildo;...
- ToyCHR interpreter and compiler for any Prolog, Duck

CHR in Functional Programming

- HaskellCHR, Duck
- CCHR (concurrent CHR), Sulzmann, Lam
- TaiChi, Boespflug

CHR in Imperative Languages

- K.U. Leuven JCHR in Java (no search), Van Weert
- K.U. Leuven CCHR in C (fast), Wuille
- JaCK (JCHR,VisualCHR,JASE) interpreter in Java, Abdennadher
- CHORD interpeter in Java (prototype), Menezes, Vitorino

CHR Implementation Projects

Support developers with environments and tools

- **CHR IDE** for Emacs and Eclipse, for Prolog and Java
- **CHR analysis tools**, e.g. confluence checker, Langbein
- **PocketCHR** for embedded systems, mobile phones, robots

Enable new application domains through non-computer hardware

Hot Application Areas in Computer Science

- tangible user interfaces, multimedia
- semantics web, multimedia
- design and configuration
- health care, pharmaceutical research
- biochemics, biocomputing, cell matabolism

Industrial Applications of CHR

- **Stock Broking** - Mike Elston, SecuritEase, New Zealand
- **Injection Mold Design** - Alan Baljeu, Cornerstone, Canada
- **Optical Network Routing** - Jonathan Weston-Dawkes, MITRE, USA
- **Rule-based Systems** - Beata Sarna-Starosta, Logic Blox, USA
- **Automated Software Testing** - Ralf Gerlich, BSSE, Germany (*)

Upcoming:

- **Rack Configuration** - Andreas Falkner, Siemens, Austria
- **Robot Configuration** - Uwe Lesta, SBS, Germany (*)

(*) Talk during CHR Working Week, Oct 09, Ulm

Optical Network Routing (c) Jonathan Weston-Dawkes, MITRE, USA

Constraint-based K-Shortest Paths

K-shortest path algorithms play an important role in network-based routing and resource allocation, but often link- and path-based QoS constraints f lter out most of generated paths. Consequently, large values of K are used to arrive at a specif ed minimum number of compliant paths.

Is there a CHR-based implementation that could generate the compliant paths directly?

Note: I have a prolog-based implementation of a recursive k shortest path algorithm as a starter, but a JCHR implementation would be faster.

Optical Network Routing (c) Jonathan Weston-Dawkes, MITRE, USA

Security-based Plan Generation

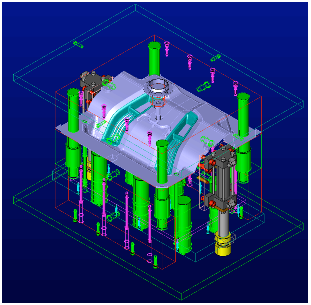
There are java (JSHOP) and prolog (GIPO) automated planning systems.

Can a CHR implementation of a security policy language (SPL) be integrated with an automated planning system to generate plans specif c to a user's privileges to system resources?

If a user's privileges permit only a summarized view of system resources, under what conditions can a plan created for the summarized system be a partial instantiation of a plan in the complete system?

Injection Mold Design (c) Alan Baljeu, Cornerstone, Canada

Knowledge-Integrated Design



Cornerstone Intelligent Software is developing constraint-based automated design technology. We have applied this technology to generate a preliminary mold design as you see on the left.

Constraint technology enables customers to integrated their standards into our knowledge base to design what they need automatically or interactively.

CORNERSTONE
INTELLIGENT SOFTWARE

Injection Mold Design (c) Alan Baljeu, Cornerstone, Canada

Design Automation

Every design answers the following questions:

- What parts to use
- What sizes are <u>available</u>
- What sizes should be used for parts
- How the parts should be arranged

These designs must satisfy both general functional requirements, and customer-specific requirements.

CORNERSTONE
INTELLIGENT SOFTWARE

Kinds of Constraints involved

- Numeric equations and inequalities (A + 2*B < 6 mm)
- Round-value requirements (B is an integer multiple of 0.1 mm)
- Symbolic (Supplier is ‘ACME’)
- Boolean logic (Plate.Thickness > 12mm **OR** Plate requires a Stiffener)
- Object selection (Primary-Pin ∈ Pins-Table **AND** Primary-Pin.Diameter = 15mm)
- Custom constraint types (the arrangement of screws should fit a grid pattern)

CORNERSTONE
INTELLIGENT SOFTWARE

Injection Mold Design (c) Alan Baljeu, Cornerstone, Canada

Design Technique

- Choose a subassembly.
- Gather constraints and parameter values related to that subassembly.
- Calculate the configuration which best <u>satisfies</u> the constraints.
- Generate the CAD model
- Proceed to design the next assembly, as required.

CORNERSTONE
INTELLIGENT SOFTWARE

Injection Mold Design (c) Alan Baljeu, Cornerstone, Canada

Implementation

- Design objects are created and managed in C#
- Parameters and constraints are passed to SWI-Prolog.
- SWI-Prolog sets up a backjumping search for the optimal satisfiable combination of constraints
- CHR rules process the constraints and generate solutions.

CORNERSTONE
INTELLIGENT SOFTWARE

Injection Mold Design (c) Alan Baljeu, Cornerstone, Canada

Collaborative Design

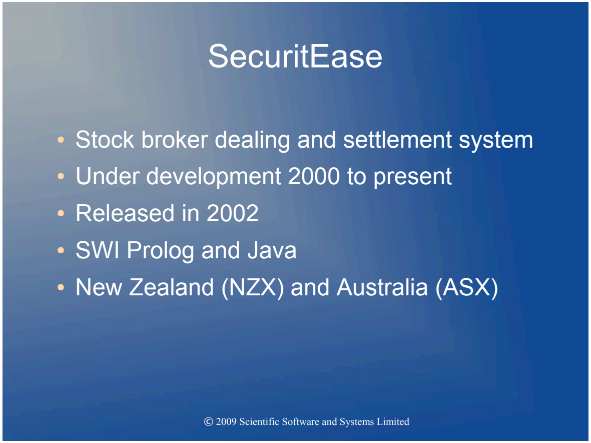
- Human designers can collaborate with the system by directly editing the generated design.
- If the designer changes one element of the design, the system will automatically update all the related elements so the design remains compliant to the constraints.

CORNERSTONE INTELLIGENT SOFTWARE

Extending the Knowledge Base

- Constraint-based systems are easily extended.
- Companies can directly add their standards to our knowledge base.
- By including their knowledge, designs can be generated which are fully compliant with all specifications, from the beginning – because they are based on the full set of requirements.

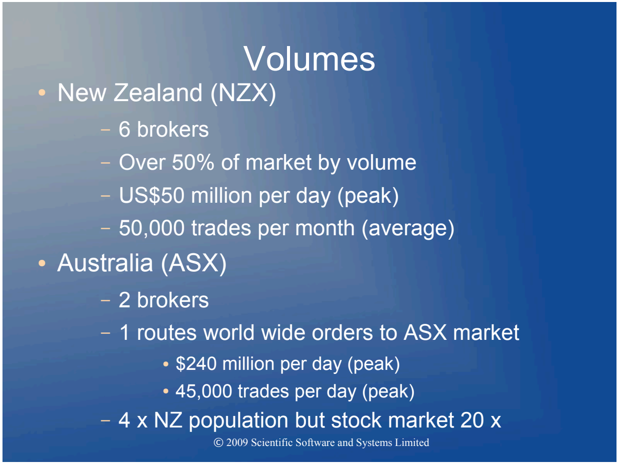
Stock Broking (c) Mike Elston, SecuritEase, New Zealand



SecuritEase

- Stock broker dealing and settlement system
- Under development 2000 to present
- Released in 2002
- SWI Prolog and Java
- New Zealand (NZX) and Australia (ASX)

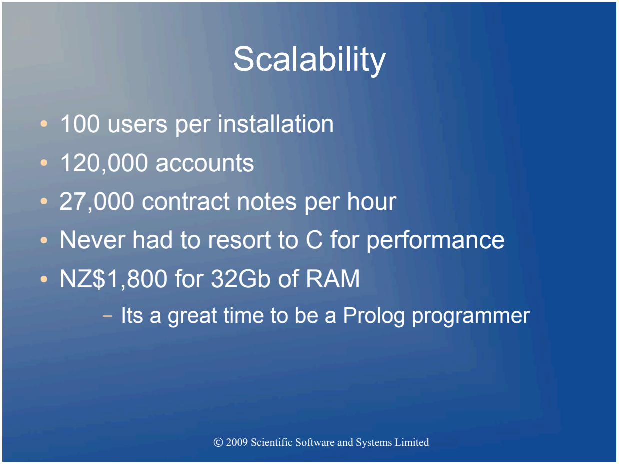
Stock Broking (c) Mike Elston, SecuritEase, New Zealand



Volumes

- New Zealand (NZX)
  - 6 brokers
  - Over 50% of market by volume
  - US$50 million per day (peak)
  - 50,000 trades per month (average)
- Australia (ASX)
  - 2 brokers
  - 1 routes world wide orders to ASX market
    - $240 million per day (peak)
    - 45,000 trades per day (peak)
  - 4 x NZ population but stock market 20 x

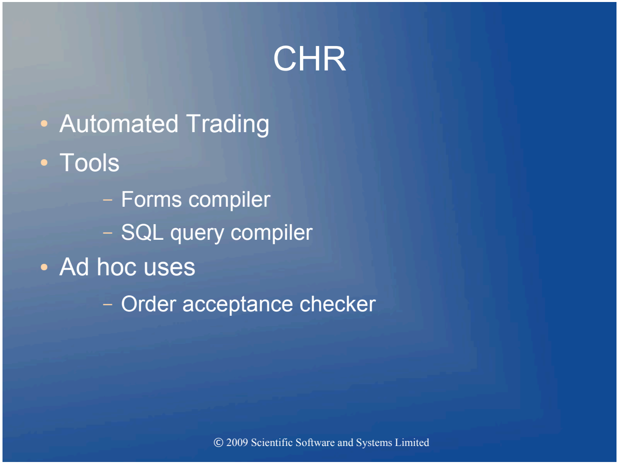
Stock Broking (c) Mike Elston, SecuritEase, New Zealand



Scalability

- 100 users per installation
- 120,000 accounts
- 27,000 contract notes per hour
- Never had to resort to C for performance
- NZ$1,800 for 32Gb of RAM
  - Its a great time to be a Prolog programmer

Stock Broking (c) Mike Elston, SecuritEase, New Zealand



CHR

- Automated Trading
- Tools
  - Forms compiler
  - SQL query compiler
- Ad hoc uses
  - Order acceptance checker

Stock Broking Mike Elston, SecuritEase, New Zealand

Internship offered!

- Two months fulltime work any time in 2010
- Return airfares paid
- Free accommodation at company apartment (near office)
- Office in Wellington, at the sea, New Zealand
- Living allowance

More information on request.

CHR Funding

- **People:** DAAD, Humboldt, DFG, EC, ERC,...
- **Projects:** DAAD, DFG, BMBF,...
- **Industrial Partners:** DAAD, Internships, Projects, Sponsors
- **University Ulm:** own local funds; SFB/Transregio Companion Technology; Graduate college Logic and Complexity;...

Details see Frank Raiser’s talk during CHR Working Week.

Conclusions

Here in Ulm, besides basic CHR **research**, we also want to offer **training** and support industrial **applications** - *the magic triangle*.

Only possible if the right people do it.

It’s you who counts!