$\mathcal{L}^2_{K,P}$-based Constraint Logic Programming

Frank Morawietz
SFB 340, A8
Universität Tübingen
Wilhelmstr. 113
72074 Tübingen
Germany
frank@sfs.nphil.uni-tuebingen.de

August 13th, 1996

Definition 1 *Let* $\mathcal{L}^2_{K,P}$ *be defined by*

K	*a set of countably many individual constants,*
P	*a set of countably many propositional constants,*
$\mathbf{X}$	*a countably infinite set of monadic second order variables ,*
$\wedge, \vee, \neg$	*logical connectives*
$\forall, \exists$	*quantifiers,*
$(,), [,]$	*brackets*
$\lhd$, $\lhd^*$, $\lhd^+$, $\prec$, $\approx$, $\in$	*relation symbols*

Terms, atomic formulae, literals and well-formed formulae are build up as usual.

We refer to $\mathcal{L}^2_{\emptyset,\emptyset}$ *simply as* $\mathcal{L}^2$*.*

Examples for $\mathcal{L}^2_{K,P}$ formulae

$$
\begin{aligned}
X \subseteq Y &\equiv \forall x[x \in X \to x \in Y] \\
sing(X) &\equiv \forall Y, Z[Y \subseteq X \to (X \subseteq Y \vee Y \subseteq Z)] \wedge \\
&\quad\ \exists Y[X \not\subseteq Y]
\end{aligned}
$$

Definition 2 *A model for* $\mathcal{L}^2_{K,P}$ *is a tuple* $\langle \mathcal{U}, \mathcal{I}, \mathcal{P}, \mathcal{D}, \mathcal{L}, \mathcal{R}_p \rangle_{p \in P}$ *with:*

$\mathcal{U}$	*a nonempty universe*
$\mathcal{I}$	*a function from* K *to* $\mathcal{U}$
$\mathcal{P}$, $\mathcal{D}$, $\mathcal{L}$	*binary relations on* $\mathcal{U}$*, which interpret* $\lhd$*,* $\lhd^*$ *and* $\prec$
$\mathcal{R}_p$	*relations of appropriate arity on* $\mathcal{U}$ *interpreting* $p \in P$*.*

A model for $\mathcal{L}^2$ *is then just* $\langle \mathcal{U}, \mathcal{P}, \mathcal{D}, \mathcal{L} \rangle$ *.*

Definition 3 *A variable assignment* α *maps each variable to an appropriate subset of* $\mathcal{U}$*. Satisfaction then is relative to such an assignment. We write*

$$M \models \phi[\alpha]$$

for satisfaction and $M \models \phi$ *then means that* M *models* ϕ *for every assignment* α*.*

Definition 4 *The n-branching tree domain* $T_n = \mathbf{n}^*$ *for* $n \in I\!N \cup \{\omega\}$ *and* $\mathbf{n} = \{i \,|\, i < n\}$ *is the tree domain in which every address has* n *successors.*

The lexicographic order ($\ll$) *on* T_n *for* $x, y, z, u, v \in \mathbf{n}^*$ *and* $a, b \in \mathbf{n}$ *is defined as:*

$x \ll y \overset{def}{\Longleftrightarrow} x \lhd^* y$ *or* $x = zau, y = zbv,$ *and* $a < b$

We define the structure of n *successor functions as*

$$\mathcal{N}_n := \langle T_n, \lhd^*, \ll, s_i \rangle_{i<n}$$

Theorem 5 (Rabin) *The monadic second-order theory of* n *successor functions is decidable for every* $n < \omega$.

Definition 6 *An $\mathcal{I}$-assignment is a total function $\alpha : VAR \to D^{\mathcal{I}}$. We call the set of all those mappings $ASS^{\mathcal{I}}$.*

Definition 7 *An interpretation $\mathcal{I}$ of the constraint language is a pair ($\mathcal{M}$,$[\![-]\!]^{\mathcal{I}}$) consisting of nonempty $\mathcal{M} = \langle \mathcal{U}, \mathcal{P}, \mathcal{D}, \mathcal{L} \rangle$ with the domain of $\mathcal{I}$ being $D^{\mathcal{I}} \subseteq \wp\mathcal{U}$ and solution mappings $[\![-]\!]^{\mathcal{I}}$ with:*

1. *$[\![-]\!]^{\mathcal{I}}$ is a function mapping every constraint ϕ to a set $[\![\phi]\!]^{\mathcal{I}}$ of $\mathcal{I}$-assignments. We call that set the solutions of ϕ in $\mathcal{I}$.*

2. *a constraint ϕ constrains only the free variables of ϕ, i.e., if $\alpha \in [\![\phi]\!]^{\mathcal{I}}$ and β is an $\mathcal{I}$-assignment witch agrees with β on the free variables, then $\beta \in [\![\phi]\!]^{\mathcal{I}}$.*

Definition 8 *The constraint language $\mathcal{CL}^2$ is a fourtuple $\langle VAR, CON, V, INT\rangle$ such that*

1. *VAR is a countable, (possibly) infinite set of (monadic second order) variables, i.e., $\mathbf{X}$*

2. *CON is a decidable set of constraints of well formed formulae (of monadic second-order logic), i.e., $\mathcal{L}^2$ formulae,*

3. *V is a computable function which assigns to every $\phi \in CON$ a finite set $V(\phi)$ of variables, called the variables constrained by ϕ or the free variables of ϕ,*

4. *INT is a nonempty set of interpretations $\mathcal{I}$.*

Definition 9 *The function* $V : CON \rightarrow VAR$ *is defined as*

$$
\begin{array}{lcl}
V(X\,r\,Y) & = & X \cup Y \\
V(\neg\phi) & = & V(\phi) \\
V(\phi * \psi) & = & V(\phi) \cup V(\psi) \\
V(\mathcal{Q}X\phi) & = & V(\phi) - \{X\}
\end{array}
$$

Definition 10 *The denotation function ($[\![-]\!]^{\mathcal{I}}$) relative to an interpretation $\mathcal{I}$ containing the model $\mathcal{M} = \langle \mathcal{U}, \mathcal{P}, \mathcal{D}, \mathcal{L} \rangle$ on $\mathcal{L}^2$-constraints is defined inductively as follows:*

$$
\begin{array}{lcl}
[\![x \lhd y]\!]^{\mathcal{I}} & = & \{\alpha \,|\, \langle \alpha(x), \alpha(y) \rangle \in \mathcal{P}\} \\
[\![x \lhd^{*} y]\!]^{\mathcal{I}} & = & \{\alpha \,|\, \langle \alpha(x), \alpha(y) \rangle \in \mathcal{D}\} \\
[\![x \prec y]\!]^{\mathcal{I}} & = & \{\alpha \,|\, \langle \alpha(x), \alpha(y) \rangle \in \mathcal{L}\} \\
[\![x \approx y]\!]^{\mathcal{I}} & = & \{\alpha \,|\, \alpha(x) = \alpha(y)\} \\
[\![x \in X]\!]^{\mathcal{I}} & = & \{\alpha \,|\, \alpha(x) \in \alpha(X)\} \\
[\![\phi \wedge \psi]\!]^{\mathcal{I}} & = & [\![\phi]\!]^{\mathcal{I}} \cap [\![\psi]\!]^{\mathcal{I}} \\
[\![\phi \vee \psi]\!]^{\mathcal{I}} & = & [\![\phi]\!]^{\mathcal{I}} \cup [\![\psi]\!]^{\mathcal{I}} \\
[\![\neg \phi]\!]^{\mathcal{I}} & = & ASS^{\mathcal{I}} - [\![\phi]\!]^{\mathcal{I}} \\
[\![\exists X \phi]\!]^{\mathcal{I}} & = & \{\alpha \,|\, \alpha_{\frac{X}{Y}} \in [\![\phi]\!]^{\mathcal{I}} \text{ \textit{for some} } Y \subseteq \mathcal{U}\} \\
[\![\forall X \phi]\!]^{\mathcal{I}} & = & \{\alpha \,|\, \alpha_{\frac{X}{Y}} \in [\![\phi]\!]^{\mathcal{I}} \text{ \textit{for all} } Y \subseteq \mathcal{U}\}
\end{array}
$$

Definition 11 *A renaming is a bijection* $\rho : VAR \to VAR$ *that is the identity except for finitely many exceptions.*

If ρ *is a renaming, we call a constraint* ϕ' *a* ρ*-variant of* ϕ *if*

$$\mathcal{V}(\phi') = \rho(\mathcal{V}(\phi)) \text{ and } [\![\phi]\!]^{\mathcal{I}} = [\![\phi']\!]^{\mathcal{I}}\rho$$

with $[\![\phi']\!]^{\mathcal{I}}\rho := \{\alpha\rho \,|\, \alpha \in [\![\phi]\!]^{\mathcal{I}}\}$ *for every interpretation* $\mathcal{I}$.

A constraint is called a variant if there exists a renaming ρ *such that it is a* ρ*-variant.*

Definition 12 *A constraint language is*

1. *closed under renaming if every constraint* ϕ *has a* ρ*-variant for every renaming* ρ*,*

2. *closed under intersection if for every two constraints* ϕ *and* ψ *there exists a constraint* θ *such that for every interpretation* $\mathcal{I}$

$$[\![\phi]\!]^{\mathcal{I}} \cap [\![\psi]\!]^{\mathcal{I}} = [\![\theta]\!]^{\mathcal{I}},$$

3. *decidable if the satisfiability of its constraints is decidable.*

Lemma 13 $\mathcal{CL}^2$ *is a decidable constraint language closed under renaming and intersection.*

Definition 14 *We can extend* $\mathcal{CL}^2$ *with a decidable set of relations* $\mathcal{R}$ *to a relational constraint language* $\mathcal{R}(\mathcal{CL}^2)$ *on the same set of variables by building the* $\mathcal{R}(\mathcal{CL}^2)$*-constraints and* $\mathcal{R}(\mathcal{CL}^2)$*-interpretations from* $\mathcal{CL}^2$*.*

Definition 15 *The $\mathcal{R}(\mathcal{CL}^2)$-constraints are defined inductively*

1. *ϕ is a constraint for all $\phi \in \mathcal{CL}^2$.*

2. *$r(X_1, \ldots, X_n)$ is a constraint for all tuples of pairwise distinct variables $X_1, \ldots, X_n$ and relation symbols r of rank n. We call this an atom.*

3. *if F and G are constraints in $\mathcal{R}(\mathcal{CL}^2)$, then so are $F \mathbin{\&} G$ and $F \rightarrow G$; this includes the empty conjunction $\emptyset$.*

$\exists X[F]$ is a constraint if X is a variable and F is an $\mathcal{R}(\mathcal{CL}^2)$ constraint.

Definition 16 *The set of free variables $\mathcal{V}$ of an $\mathcal{R}(\mathcal{CL}^2)$ constraint is again defined inductively:*

1. $\mathcal{V}(\phi) := V(\phi)$ *if* ϕ *is an* $\mathcal{CL}^2$*-constraint*

2. $\mathcal{V}(r(X_1, \ldots, X_n)) := \{X_1, \ldots, X_n\}$

3. $\mathcal{V}(\emptyset) := \emptyset$

4. $\mathcal{V}(F \,\&\, G) := \mathcal{V}(F) \cup \mathcal{V}(G)$

5. $\mathcal{V}(F \to G) := \mathcal{V}(F) \cup \mathcal{V}(G)$

6. $\mathcal{V}(\exists X[F]) := \mathcal{V}(F) - \{X\}$

Definition 17 *An $\mathcal{R}(\mathcal{CL}^2)$ interpretation is the extension $\mathcal{A}$ of the interpretation $\mathcal{I}$ to include the new relational symbols by choosing relations $r^{\mathcal{A}}$ with appropriate arity on $\mathcal{U}$ for each $r \in \mathcal{R}$, setting $D^{\mathcal{A}}$ to $D^{\mathcal{I}}$ and extending the solution mapping as follows:*

1. $[\![\phi]\!]^{\mathcal{A}} := [\![\phi]\!]^{\mathcal{I}}$ *if ϕ is an $\mathcal{CL}^2$-constraint*

2. $[\![r(X_1, \ldots, X_n)]\!]^{\mathcal{A}} := \{\alpha \in ASS^{\mathcal{A}} \mid \alpha(X_1, \ldots, X_n) \in r^{\mathcal{A}}\}$

3. $[\![\emptyset]\!]^{\mathcal{A}} := ASS^{\mathcal{A}}$

4. $[\![F \,\&\, G]\!]^{\mathcal{A}} := [\![F]\!]^{\mathcal{A}} \cap [\![G]\!]^{\mathcal{A}}$

5. $[\![F \to G]\!]^{\mathcal{A}} := (ASS^{\mathcal{A}} - [\![F]\!]^{\mathcal{A}}) \cup [\![G]\!]^{\mathcal{A}}$

$$
\begin{aligned}
[\![\exists X[F]]\!]^{\mathcal{A}} := \{\alpha \in ASS^{\mathcal{A}} \mid\ & \exists\beta \in [\![F]\!]^{\mathcal{A}}\,\& \\
& \forall Y \in \mathcal{V}(F)\, Y = X \vee \\
& \alpha(X) = \alpha(Y)\}
\end{aligned}
$$

Definition 18 *A definite clause specification $\mathcal{S}$ is a set of definite clauses of the form*

$$B \leftarrow \phi \,\&\, A_1 \,\&\, \ldots \,\&\, A_n$$

where B and the A_i are atoms, $n \geq 0$ and ϕ is a $\mathcal{CL}^2$-constraint.

Theorem 19 (H&S prop. 4.4) *Let $\mathcal{S}$ be a definite clause specification in $\mathcal{R}(\mathcal{CL}^2)$ and $\mathcal{I}$ be an $\mathcal{CL}^2$-interpretation. Then we can inductively define a chain of $\mathcal{R}(\mathcal{CL}^2)$-interpretations $\mathcal{A}_i$ extending $\mathcal{I}$ by*

1. $r^{\mathcal{A}_0} := \emptyset$

2. $$r^{\mathcal{A}_{i+1}} := \{\alpha(X_1, \ldots, X_n) \mid (r(X_1, \ldots, X_n) \leftarrow Body) \in \mathcal{S} \ \& \ \alpha \in [\![Body]\!]^{\mathcal{A}_i}\}$$

The least model of $\mathcal{S}$ extending $\mathcal{I}$ is the union

$$\mathcal{A} := \bigcup_{i \geq 0} \mathcal{A}_i.$$

Definition 20 *A goal is a possibly empty conjunction of* $\mathcal{CL}^2$*-constraints and* $\mathcal{R}(\mathcal{CL}^2)$*-atoms.*

An observation is an implication $\phi \to G$ *consisting of an* $\mathcal{CL}^2$*-constraint* ϕ *and a goal* G*.*

Theorem 21 (H&S prop. 4.5) *Let $\mathcal{S}$ be a definite clause specification. Then an observation is valid in every model of $\mathcal{S}$ iff it is valid in every minimal model of $\mathcal{S}$.*

Definition 22 *Let $\mathcal{M}$ be a definite specification. An $\mathcal{M}$-answer of a goal G is a satisfiable $\mathcal{L}$-constraint ϕ such that the observation $\phi \rightarrow G$ is valid in every model of $\mathcal{M}$.*

A set Φ of $\mathcal{M}$-answers is complete if $[\![\Phi]\!]^{\mathcal{A}}_{\mathcal{V}(G)} = [\![G]\!]^{\mathcal{A}}_{\mathcal{V}(G)}$ for every minimal model $\mathcal{A}$ of $\mathcal{M}$.

Definition 23 $(\mathcal{S}, V)$-*goal reduction* ($\xrightarrow{r}_{\mathcal{S},V}$) *is defined to be*

$$A \,\&\, G \xrightarrow{r}_{\mathcal{S},V} F \,\&\, G$$

if $A \leftarrow F$ *is a variant of a clause in* $\mathcal{S}$ *such that* $(V \cup \mathcal{V}(G)) \cap \mathcal{V}(F) \subseteq \mathcal{V}(A)$.

Theorem 24 (Soundness, H&S prop. 5.1) *If* $G \xrightarrow{r}_{\mathcal{S},V} F$, *then* $[\![F]\!]^{\mathcal{A}} \subseteq [\![G]\!]^{\mathcal{A}}$ *for every model* $\mathcal{A}$ *of* $\mathcal{S}$.

Theorem 25 (H&S theo. 5.2) *Let $\mathcal{A}$ be a minimal model of $\mathcal{S}$, G be a goal, A be an atom in G and $\alpha \in [\![G]\!]_V^{\mathcal{A}}$. Then there exists a clause C in $\mathcal{S}$ such that*

1. *$(\mathcal{S}, V)$-goal reduction of G on A using a variant of C is possible,*

2. *if G_1 is obtained from G by $(\mathcal{S}, V)$-goal reduction on A using a variant of C then $\alpha \in [\![G_1]\!]_V^{\mathcal{A}}$ and the complexity of the goal is reduced.*

Corollary 26 (Completeness, H&S cor. 5.3) *Let $\mathcal{A}$ be a minimal model of $\mathcal{S}$, G be a goal, A be an atom in G and $\alpha \in [\![G]\!]^{\mathcal{A}}_{V}$. Then there exists an $\mathcal{S}$-answer ϕ of G such that $G \xrightarrow{r}{}^{*}_{\mathcal{S},V} \phi$ and $\alpha \in [\![\phi]\!]^{\mathcal{A}}_{V}$.*

Definition 27 *The inference rule for V-constraint solving ($\xrightarrow{c}_V$) is defined as follows. Let ϕ, ψ, θ be $\mathcal{CL}^2$-constraints.*

$$\phi \,\&\, \psi \,\&\, G \xrightarrow{c}_V \theta \,\&\, G$$

if $\phi \,\&\, \psi \sim_{V \cup \mathcal{V}(G)} \theta$.

Lemma 28 (H&S prop. 5.4) *Let G be a goal and $G \xrightarrow{c}_V G'$. Then $[\![G]\!]_V^{\mathcal{A}} = [\![G']\!]_V^{\mathcal{A}}$ for every interpretation $\mathcal{A}$ of $\mathcal{R}(\mathcal{CL}^2)$ and the complexity of the new goal equals the complexity of the old goal.*

The Constraint Solver MoSeL

- MoSeL is an $\mathcal{L}^2$ formulae to tree automata compiler (Morawietz and Cornell 1996)
- basis for MoSeL is a reformulation of the decidability proof on finite trees
- the reachability construction (Rabin 1970)
- conjunction, disjunction, negation, quantification
- encoding of relations
- minimization (Gécseg and Steinby 1984)
- problems

$\underline{X \subseteq Y}$

$$
\begin{aligned}
\text{States} &: \{q_0\} \\
\text{Final States} &: \{q_0\} \\
\text{Initial State} &: q_0 \\
\text{Transitions} &: \alpha(00)(q_0)(q_0) = q_0 \\
&\quad \alpha(01)(q_0)(q_0) = q_0 \\
&\quad \alpha(11)(q_0)(q_0) = q_0
\end{aligned}
$$

$\underline{sing(X)}$

$$
\begin{aligned}
\text{States} &: \{q_0, q_1\} \\
\text{Final States} &: \{q_1\} \\
\text{Initial State} &: q_0 \\
\text{Transitions} &: \alpha(0)(q_0)(q_0) = q_0 \\
&\quad \alpha(0)(q_0)(q_1) = q_1 \\
&\quad \alpha(0)(q_1)(q_0) = q_1 \\
&\quad \alpha(1)(q_0)(q_0) = q_1
\end{aligned}
$$

$\underline{X \subseteq Y \wedge sing(X)}$

$$
\begin{aligned}
\text{States} &: \{(q_0, q_0), (q_0, q_1)\} \\
\text{Final States} &: \{(q_0, q_1)\} \\
\text{Initial State} &: (q_0, q_0) \\
\text{Transitions} &: \alpha(00)((q_0, q_0))((q_0, q_0)) = (q_0, q_0) \\
&\quad \alpha(00)((q_0, q_0))((q_0, q_1)) = (q_0, q_1) \\
&\quad \alpha(00)((q_0, q_1))((q_0, q_0)) = (q_0, q_1) \\
&\quad \alpha(01)((q_0, q_0))((q_0, q_0)) = (q_0, q_0) \\
&\quad \alpha(01)((q_0, q_0))((q_0, q_1)) = (q_0, q_1) \\
&\quad \alpha(01)((q_0, q_1))((q_0, q_0)) = (q_0, q_1) \\
&\quad \alpha(11)((q_0, q_0))((q_0, q_0)) = (q_0, q_1)
\end{aligned}
$$

Example for Nondeterminism through Projection

$$\begin{aligned}\alpha(00)(q_0)(q_1) &= q_1\\ \alpha(01)(q_0)(q_1) &= q_0\end{aligned}$$

$$\alpha(0)(q_0)(q_1) = \{q_0, q_1\}$$

The Advantages of MoSeL

- it obviously represents a correct implementation of the necessary processes;

- it can be used incrementally since the nature of the proof allows us to view a tree automaton as a constraint store where the conjunction with a new constraint yields a new store;

- emptiness, i.e., satisfiability, is trivially decidable after automata construction;

- we can even create a library of constraints to avoid recomputation and view this as an extension of the signature of $\mathcal{L}^2$;

- we can define what it means for an $\mathcal{CL}^2$ constraint to be normal.

Interpreter

Definition 29 *A normal $\mathcal{CL}^2$-constraint is a formula whose corresponding minimized tree automaton recognizes a language which is non empty, i.e., has a final state.*

Lemma 30 *For every pair of two normal $\mathcal{CL}^2$-constraints we can compute their conjunction as a new $\mathcal{CL}^2$-constraint and decide whether it is normal.*

Definition 31 *A normal goal G is a goal which contains only one normal $\mathcal{CL}^2$-constraint.*

Definition 32 *A normal definite clause specification $\mathcal{S}$ is a definite clause specification such that the constraints in the clauses of $\mathcal{S}$ are normal.*

Lemma 33 *Every definite clause specification can be converted to a normal definite clause specification.*

Extensions to the Interpreter

- type discipline for variables (Höhfeld and Smolka 1988)
- constraint propagation (Le Provost and Wallace 1993)
- library, storage of computations
- entailment check, delay of goals (Saraswat (1993), Smolka (1993))
- program transformations (Beeri and Ramakrishnan (1987), Ramakrishnan (1991), Strzalkowski (1991), Minnen and Gerdemann (1993))
- ...

Pure $\mathcal{L}^2_{K,P}$ Parsing

$$
\begin{aligned}
parse(In, X) \;&\leftarrow \\
&\{gram(In, X)\}
\end{aligned}
$$

$$
\begin{aligned}
yield(S, T) \;&\equiv\; \forall x[(x \in T \wedge \neg\exists y[x \lhd y] \wedge \\
&\qquad\quad \neg ephon(x)) \rightarrow x \in S] \wedge \\
&\quad\; \neg\exists z[z \in S \wedge z \notin T] \\
gram(In, X) \;&\equiv\; yield(In, Tree) \wedge gram(Tree)
\end{aligned}
$$

Lists in $\mathcal{R}(\mathcal{CL}^2)$

$$
\begin{array}{rcl}
min(x, X) & \equiv & x \in X \wedge \forall y[(y \in X \wedge y \not\approx x) \rightarrow x \prec y] \\
max(x, X) & \equiv & x \in X \wedge \forall y[(y \in X \wedge y \not\approx x) \rightarrow y \prec x] \\
elist(X) & \equiv & \neg \exists x[x \in X] \\
nelist(X) & \equiv & \exists x, y[min(x, X) \wedge max(y, X)] \wedge \\
 & & \forall x, y[\neg min(x, X) \wedge \neg min(y, X) \wedge \\
 & & \qquad \neg max(x, X) \wedge \neg max(y, X) \wedge \\
 & & \qquad (x \prec y \vee y \prec x)] \\
list(X) & \equiv & elist(X) \vee nelist(X) \\
hd(X, x) & \equiv & min(x, X) \\
tl(X, Y) & \equiv & Y \subseteq X \wedge \\
 & & \forall x[(\neg min(x, X) \wedge x \in X) \rightarrow x \in Y]
\end{array}
$$

$$
\begin{aligned}
append(X, Y, Z) \;\; & \leftarrow \\
& \{elist(X) \wedge Y \approx Z\}
\end{aligned}
$$

$$
\begin{aligned}
append(X, Y, Z) \;\; & \leftarrow \\
& \{hd(X, x) \wedge hd(Z, x) \wedge \\
& \;\; tl(X, U) \wedge tl(Z, V)\} \quad \& \\
& append(U, Y, V)
\end{aligned}
$$

Johnson (1995) I

$$
\begin{array}{lll}
parse(Words, Tree) & \leftarrow & \\
& \{list(Words)\} & \& \\
& xbar(Tree) & \& \\
& ecp(Tree) & \& \\
& yield(Words, Tree) &
\end{array}
$$

$$
\begin{array}{rcl}
root(X, x) & \equiv & x \in X \wedge \\
& & \forall y[y \in X \rightarrow x \lhd^* y] \\
dtrs(x, y, z) & \equiv & x \lhd y \wedge x \lhd z \wedge y \prec z \\
subtree(X, y, Y) & \equiv & \forall x[(x \in X \wedge y \lhd^* x) \rightarrow \\
& & \quad x \in Y] \\
strs(X, x, Y, y, Z, z) & \equiv & root(X, x) \wedge dtrs(x, y, z) \wedge \\
& & subtree(X, y, Y) \wedge \\
& & subtree(X, z, Z)
\end{array}
$$

Johnson (1995) II

$$
\begin{array}{ll}
xbar(X) & \leftarrow \\
 & \{elist(X)\}
\end{array}
$$

$$
\begin{array}{ll}
xbar(X) & \leftarrow \\
 & \{sing(X)\} \quad \& \\
 & lexicon(X)
\end{array}
$$

$$
\begin{array}{lll}
xbar(X) & \leftarrow & \\
 & \{strs(X, x, Y, y, Z, z)\} & \& \\
 & local\_xbar(x, y, z) & \& \\
 & xbar(Y) & \& \\
 & xbar(Z) &
\end{array}
$$

Fong (1991)

$$\begin{array}{rcl} max(x) & \equiv & x \in NP \vee x \in VP \vee \ldots \\ proj(x,y) & \equiv & (x \in NP \leftrightarrow y \in N1) \vee \\ & & (x \in VP \leftrightarrow y \in V1) \vee \ldots \\ daughters(x,X) & \equiv & \forall y[x \lhd y \rightarrow y \in X] \wedge list(X) \end{array}$$

$$\begin{array}{ll} xp(x) & \leftarrow \\ & \{max(x) \wedge proj(x,y) \wedge daughters(x,X) \wedge \\ & \ hd(X,y) \wedge tl(X,Z)\} \qquad \& \\ & x1(y) \qquad \& \\ & specs(y,Z) \end{array}$$

The Lexicon

$$
\begin{aligned}
lex(x) \;&\leftarrow \\
&\{See(x) \wedge V(x) \wedge \ldots\}
\end{aligned}
$$

$$
\begin{aligned}
lex(x) \;&\leftarrow \\
&\{Sleep(x) \wedge V(x) \wedge \ldots\}
\end{aligned}
$$

$$
\begin{aligned}
lex(x) \;&\leftarrow \\
&\{Lena(x) \wedge N(x) \wedge \ldots\}
\end{aligned}
$$

$$\vdots$$

Nonlocal Constraints

$$
\begin{array}{lll}
parse(In, X) & \leftarrow & \\
 & \{yield(In, X)\} & \& \\
 & \theta\text{-}theory(X) & \& \\
 & xbar(X) & \& \\
 & case(X) & \& \\
 & move\text{-}\alpha(X) & \& \\
 & lexicon(In) & \& \\
 & \vdots &
\end{array}
$$

$$
\begin{array}{ll}
lexicon(X) & \leftarrow \\
 & \{elist(X)\}
\end{array}
$$

$$
\begin{array}{lll}
lexicon(X) & \leftarrow & \\
 & \{hd(X, x) \wedge tl(X, Y)\} & \& \\
 & lex(x) & \& \\
 & lexicon(Y) &
\end{array}
$$

$$
\begin{array}{ll}
lexicon(X) & \leftarrow \\
 & \{\forall x[x \in X \rightarrow lex(x)]\}
\end{array}
$$

Quantification in the Constraint Language

$$subtrees(X, Y, Z) \;\equiv\; \exists x, y, z[strs(X, x, Y, y, Z, z)]$$

$$\begin{array}{lll}
s(X) & \leftarrow & \\
& \{subtrees(X, Y, Z) \wedge & \\
& \;\exists z[z \in Z \wedge z \in Trace \wedge feq(z, Y)]\} & \& \\
& np_{wh}(Y) & \& \\
& vp(Z) &
\end{array}$$

Datatypes

$$
\begin{array}{rcl}
cmp(X) & \equiv & path(X) \wedge \\
& & \forall x, y[(x \in X \wedge y \in X) \rightarrow feq(x,y)] \wedge \\
& & \forall x, x' \exists y \forall z[(x \in X \wedge x' \in X \wedge x \lhd x') \rightarrow \\
& & \qquad\qquad (x' \notin Adj \wedge x \lhd y \wedge y \not\approx x' \wedge \\
& & \qquad\qquad\; y \in Adj \wedge \\
& & \qquad\qquad (x \lhd z \rightarrow (z \approx x' \vee z \approx y)))] \\
cat(X) & \equiv & cmp(X) \wedge \\
& & \forall Y[(X \subseteq Y \wedge Y \not\subseteq X) \rightarrow \\
& & \qquad \neg cmp(Y)]
\end{array}
$$

Summary

- We can give a procedural interpretation

- Is $\mathcal{CL}^2$ compact?

- Do we gain expressive power?

- How does a $\mathcal{R}(\mathcal{CL}^2)$-parser which really uses the power of $\mathcal{L}^2_{K,P}$ look?

- What happens with the semantics when we extend $\mathcal{R}(\mathcal{CL}^2)$?

Beeri, C and Ramakrishnan, R. (1987). On the Power of Magic, *Proceedings of the 6th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*.

Fong, Sandiway (1991). *Computational Properties of Principle–Based Grammatical Theories*, PhD thesis, MIT.

Gécseg, F. and Steinby, M. (1984). *Tree Automata*, Akadémiai Kiadó, Budapest.

Höhfeld, Markus and Smolka, Gerd (1988). Definite Relations over Constraint Languages, *LILOG Report 53*, IBM Deutschland, Stuttgart, Germany.

Johnson, Mark (1995). Constraint-Based Natural Language Parsing, ESSLLI '95 Barcelona notes.

Le Provost, Thierry and Wallace, Mark (1993). Generalized Constraint Propagation over the CLP scheme, *Journal of Logic Programming* **16**: 319–359.

Minnen, Guido and Gerdemann, Dale D. (1993). Direct Automated Inversion of Logic Grammars, *SFB-Report 35*, SFB 340, University of Tübingen.

Morawietz, Frank and Cornell, Thomas L. (1996). MoSeL: A Tool for Monadic Second Order Logic on Finite Trees, *Technical Report*, University of Tuebingen, SFB 340, A8.

Rabin, Michael O. (1970). Weakly definable relations and special automata, *in* Y. Bar-Hillel (ed.), *Mathematical Logic and Foundations of Set Theory*, North-Holland, Amsterdam, pp. 1–23.

Ramakrishnan, Raghu (1991). Magic Templates: A Spell-binding Approach to Logic Programs, *Journal of Logic Programming* **11**: 189–216.

Saraswat, Vijay A. (1993). *Concurrent Constraint Programming*, MIT Press, Cambridge, Massachusetts.

Smolka, Gerd (1993). Residuation and Guarded Rules for Constraint Logic Programming, *in* F. Benhamou and A. Colmerauer (eds.), *Constraint Logic Programming: Selected Research*, MIT Press, chapter 21, pp. 405–419.

Strzalkowski, T. (1991). A General Computational Method for Grammar Inversion, *Proceedings of the Reversible Grammar Workshop*, Berkeley, CA.