Constraint Logic Programming for Hedges: A Semantic Reconstruction

Besik Dundua

DCC-FC & LIACC, University of Porto, Portugal

VIAM, Ivane Javakhishvili Tbilisi State University, Georgia

Joint work with Mário Florido, Temur Kutsia and Mircea Marin

What is CLP(H)

- CLP(H): Constraint Logic Programming over hedges.

What is CLP(H)

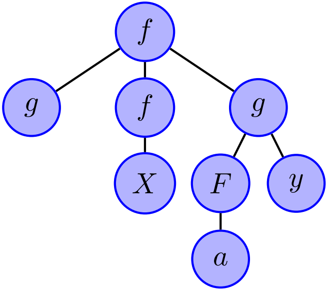
- CLP(H): Constraint Logic Programming over hedges.
- Hedges: finite sequences of unranked terms and hedge variables.

What is CLP(H)

- CLP(H): Constraint Logic Programming over hedges.
- Hedges: finite sequences of unranked terms and hedge variables.
- Unranked terms: function symbols have no fixed arity.

Unranked Term: Example

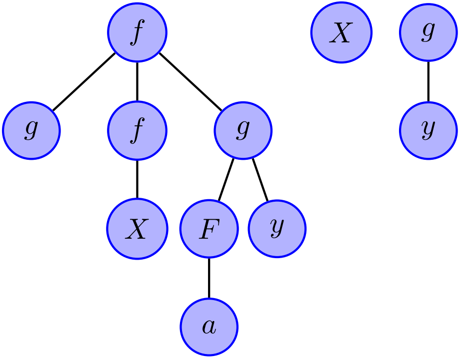
$$f(g, f(X), g(F(a), y))$$



- Different occurrences of the same function symbol may have different number of arguments.
- Variables: X for hedges, y for terms, F for function symbols.

Hedge: Example

$$f(g, f(X), g(F(a), y)), \quad X, \quad g(y)$$



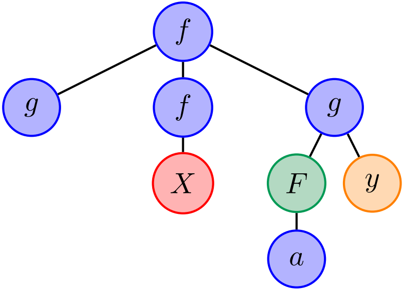
- Finite sequences of unranked terms and hedge variables.

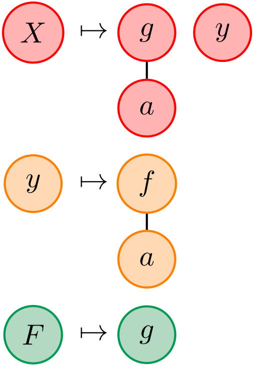
Variables

- Term variables – can be instantiated by individual terms.
- Hedge variables – can be instantiated by hedges.
- Function variables – can be instantiated by function symbols.

Variable Instantiation: Example

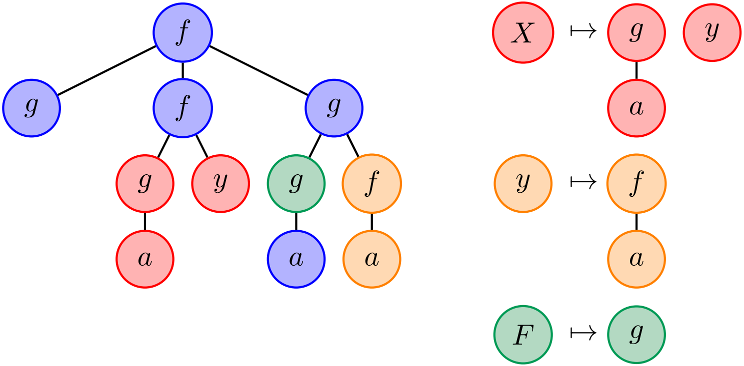
$f(g, f(X), g(F(a), y))$ $\{X \mapsto (g(a), y), y \mapsto f(a), F \mapsto g\}$





Variable Instantiation: Example

$f(g, f(X), g(F(a), y))$ $\qquad$ $\{X \mapsto (g(a), y), y \mapsto f(a), F \mapsto g\}$



CLP(H) Programs

- Three kinds of variables give flexibility of term traversal.
- It helps to write short, yet quite clear and intuitive code.

CLP(H) Programs

- Three kinds of variables give flexibility of term traversal.
- It helps to write short, yet quite clear and intuitive code.

Example (Rewriting)

$$rewrite(x, y) \leftarrow rule(x, y).$$
$$rewrite(F(X, x, Y), F(X, y, Y)) \leftarrow rewrite(x, y).$$

$$rule(x, y) \leftarrow \ldots$$
$$\ldots$$

CLP(H) Programs

- Hedges may be constrained with regular hedge languages.

CLP(H) Programs

- Hedges may be constrained with regular hedge languages.

Example (Rewriting)

$$rewrite(x, y) \leftarrow rule(x, y).$$
$$rewrite(F(X, x, Y), F(X, y, Y)) \leftarrow rewrite(x, y).$$

$$rule(f(X), f(b, X, b)) \leftarrow X \text{ in } a^*.$$
$$\ldots$$

In This Talk

- Semantics of CLP(H).
- How to solve constraints.
- Special fragments.

Let's Get a Bit Formal

The alphabet contains

- term, hedge and function variables,
- unranked function symbols,
- ranked predicate symbols,
- true, false, $\doteq$, in,
- regular operators,
- logical connectives.

Let's Get a Bit Formal

- Terms are term variables or compound terms:

$$t ::= x \mid f(H) \mid F(H).$$

- Hedge elements are terms or hedge variables:

$$h ::= t \mid X.$$

- Hedges are finite sequences of hedge elements:

$$H ::= h_1, \ldots, h_n, \qquad n \geq 0.$$

Notation:

x: term variable	f: function symbol
F: function variable	X: hedge variable

Let's Get a Bit Formal

- Regular hedge expressions:

$$
\begin{aligned}
\mathsf{R} ::= \;& \mathsf{eps} && \text{(empty hedge expression)} \\
\mid\;& \mathsf{R} \cdot \mathsf{R} && \text{(concatenation)} \\
\mid\;& \mathsf{R} + \mathsf{R} && \text{(choice)} \\
\mid\;& \mathsf{R}^* && \text{(repetition)} \\
\mid\;& f(\mathsf{R}) && \text{(function application)}
\end{aligned}
$$

Let's Get a Bit Formal

- Regular hedge expressions:

$$
\begin{aligned}
\mathsf{R} ::= \quad & \mathsf{eps} && \text{(empty hedge expression)} \\
| \ & \mathsf{R} \cdot \mathsf{R} && \text{(concatenation)} \\
| \ & \mathsf{R} + \mathsf{R} && \text{(choice)} \\
| \ & \mathsf{R}^* && \text{(repetition)} \\
| \ & f(\mathsf{R}) && \text{(function application)}
\end{aligned}
$$

Example

- $f((a(\mathsf{eps}) + b(\mathsf{eps}))^*) \cdot c(\mathsf{eps})^*$ is a regular hedge expression.
- For simplicity, it is written as $f((a+b)^*) \cdot c^*$.

More Notions

Primitive constraints:

- Equalities: $t_1 \doteq t_2$.
- Membership atoms: H in R.

More Notions

Primitive constraints:

- Equalities: $t_1 \doteq t_2$.
- Membership atoms: H in R.

Example

- Equational primitive constraints:
  - $f(X, a) = f(a, X)$.
  - $f(X, F(Y), Z) \doteq f(a, x, f(X))$.

More Notions

Primitive constraints:

- Equalities: $t_1 \doteq t_2$.
- Membership atoms: H in R.

Example

- Equational primitive constraints:
  - $f(X, a) = f(a, X)$.
  - $f(X, F(Y), Z) \doteq f(a, x, f(X))$.
- Membership primitive constraints:
  - $(f(a, a), X, a)$ in $f((a + b)^*) \cdot c^*$.
  - X in $b^* \cdot a$.

More Notions

- Atoms: $p(t_1, \ldots, t_n)$, where p is an n-ary predicate symbol.
- Literal: An atom or a primitive constraint.
- Formulas are defined as usual.

More Notions

- Atoms: $p(t_1, \ldots, t_n)$, where p is an n-ary predicate symbol.
- Literal: An atom or a primitive constraint.
- Formulas are defined as usual.

Example

- $rewrite(F(X, x, Y), F(X, y, Y))$ is an atom.

More Notions

- Constraint: A formula built over true, false, and primitive constraints.
- We work with constraints in disjunctive normal form.

More Notions

- Constraint: A formula built over true, false, and primitive constraints.
- We work with constraints in disjunctive normal form.
- CLP program: A finite set of rules of the form $\forall(L_1 \wedge \cdots \wedge L_n \to A)$, written as

  $$A \leftarrow L_1, \ldots, L_n,$$

  where A is an atom and the L's are literals.
- Goal: A formula of the form $\exists(L_1 \wedge \cdots \wedge L_n)$, $n \geq 0$, written as

  $$\leftarrow L_1, \ldots, L_n.$$

CLP(H) Programs and Goals: Examples

- Program for removing duplicate arguments from a term:

$$\begin{aligned} & remove\_duplicates(F(X, x, Y, x, Z), y) \leftarrow \\ & \qquad remove\_duplicates(F(X, x, Y, Z), y). \\ & remove\_duplicates(x, x). \end{aligned}$$

- Goal: Find a term, obtained by removing duplicate arguments from $f(a, g(b), g(b), a, c)$:

$$\leftarrow remove\_duplicates(f(a, g(b), g(b), a, c), y).$$

CLP(H) Programs and Goals: Examples

- A program that implements the rewriting mechanism, together with a rule to perform rewritings of the form $f \to f(b,b)$, $f(a) \to f(b,a,b)$, $f(a,a) \to f(b,a,a,b)$, etc.

$$rewrite(x,y) \leftarrow rule(x,y).$$
$$rewrite(F(X,x,Y),F(X,y,Y)) \leftarrow rewrite(x,y).$$

$$rule(f(X),f(b,X,b)) \leftarrow X \text{ in } a^*.$$

- Goal: Find a term that rewrites to $f(a,f(b,f(b,a,a,b)))$:

$$\leftarrow rewrite(x,f(a,f(b,f(b,a,a,b)))).$$

Declarative Semantics

- A structure for our language: $\mathfrak{S} := \langle D, I \rangle$.
- D: a non-empty domain.
- I: an interpretation function, mapping
  - each function symbol f to a function $I(f) : D^* \to D$,
  - each n-ary predicate symbol p to an n-ary relation $I(p) \subseteq D^n$.
- A variable assignment for $\mathfrak{S}$: a function that maps
  - term variables to elements of D,
  - hedge variables to elements of D^*,
  - function variables to functions from D^* to D.

Declarative Semantics

Interpretation of syntactic categories with respect to a structure $\mathfrak{S} = \langle D, I \rangle$ and a variable assignment σ.

- Terms are interpreted as elements of D:

$$[\![v]\!]_{\mathfrak{S},\sigma} := \sigma(v),$$
$$[\![f(H)]\!]_{\mathfrak{S},\sigma} := I(f)([\![H]\!]_{\mathfrak{S},\sigma}),$$
$$[\![F(H)]\!]_{\mathfrak{S},\sigma} := \sigma(F)([\![H]\!]_{\mathfrak{S},\sigma}).$$

- Hedges are interpreted as elements of D^*:

$$[\![(H_1, \ldots, H_n)]\!]_{\mathfrak{S},\sigma} := ([\![H_1]\!]_{\mathfrak{S},\sigma}, \ldots, [\![H_n]\!]_{\mathfrak{S},\sigma}),$$

Declarative Semantics

Interpretation of syntactic categories with respect to a structure $\mathfrak{S} = \langle D, I \rangle$ and a variable assignment σ.

- Regular expressions are interpreted as (regular) subsets of D^*: (σ has no effect and is omitted.)

$$
\begin{aligned}
&[\![\mathsf{eps}]\!]_{\mathfrak{S}} := \{\epsilon\},\\
&[\![\mathsf{R}_1 \cdot \mathsf{R}_2]\!]_{\mathfrak{S}} := \{(H_1, H_2) \mid H_1 \in [\![\mathsf{R}_1]\!]_{\mathfrak{S}}, H_2 \in [\![\mathsf{R}_2]\!]_{\mathfrak{S}}\},\\
&[\![\mathsf{R}_1 + \mathsf{R}_2]\!]_{\mathfrak{S}} := [\![\mathsf{R}_1]\!]_{\mathfrak{S}} \cup [\![\mathsf{R}_2]\!]_{\mathfrak{S}},\\
&[\![\mathsf{R}^*]\!]_{\mathfrak{S}} := [\![\mathsf{R}]\!]_{\mathfrak{S}}^*.\\
&[\![f(\mathsf{R})]\!]_{\mathfrak{S}} := \{I(f)(H) \mid H \in [\![\mathsf{R}]\!]_{\mathfrak{S}}\},
\end{aligned}
$$

Declarative Semantics

Interpretation of syntactic categories with respect to a structure $\mathfrak{S} = \langle D, I \rangle$ and a variable assignment σ.

- Primitive equational constraints are interpreted as equality:

$$\mathfrak{S} \models_\sigma t_1 \doteq t_2 \text{ iff } [\![t_1]\!]_{\mathfrak{S},\sigma} = [\![t_2]\!]_{\mathfrak{S},\sigma}.$$

- Primitive membership constraints are interpreted as set membership:

$$\mathfrak{S} \models_\sigma H \text{ in } \mathsf{R} \text{ iff } [\![H]\!]_{\mathfrak{S},\sigma} \in [\![\mathsf{R}]\!]_{\mathfrak{S}}.$$

- Other formulas are interpreted in the standard way.

Declarative Semantics

Intended structure: $\mathfrak{I} = \langle D, I \rangle$, where

- D is the set of ground terms,
- I defined for every f by $I(f)(H) = f(H)$.

Declarative Semantics

Intended structure: $\mathfrak{I} = \langle D, I \rangle$, where

- D is the set of ground terms,
- I defined for every f by $I(f)(H) = f(H)$.

Intended interpretation of a program P: a subset of the Herbrand basis of P.

Declarative Semantics

Notation:

- $\mathfrak{S} \models A$: $\mathfrak{S}$ is a model of A.
- $\models A$: Any structure is a model of A.
- $P \models G$ if G is a goal which holds in every model of the program P.

Facts:

1. Every program P has a least intended model, denoted $lm(P)$.
2. For every program P and goal G, $P \models G$ iff $lm(P) \models G$.

Constraints

- $\mathcal{K}$ stands for conjunction of primitive constraints.
- $\mathcal{K}$ in the solved form, example:
  - $x \doteq f(a, X) \wedge Y \doteq (a, f(b), X) \wedge X$ in $f(a)^* \cdot b$.

Constraints

- $\mathcal{K}$ stands for conjunction of primitive constraints.
- $\mathcal{K}$ in the solved form, example:
  - $x \doteq f(a, X) \wedge Y \doteq (a, f(b), X) \wedge X$ in $f(a)^* \cdot b$.
- $\mathcal{K}$ not in the solved form, examples:
  1. $x \doteq f(a, X) \wedge (Y, a) \doteq (a, f(b), X) \wedge X$ in $f(a)^* \cdot b$.

Constraints

- $\mathcal{K}$ stands for conjunction of primitive constraints.
- $\mathcal{K}$ in the solved form, example:
  - $x \doteq f(a, X) \wedge Y \doteq (a, f(b), X) \wedge X$ in $f(a)^* \cdot b$.
- $\mathcal{K}$ not in the solved form, examples:
  1. $x \doteq f(a, X) \wedge (Y, a) \doteq (a, f(b), X) \wedge X$ in $f(a)^* \cdot b$.
  2. $x \doteq f(a, X) \wedge (Y, a, f(b)) \doteq (a, f(b), Y) \wedge X$ in $f(a)^* \cdot b$.

Constraints

- $\mathcal{K}$ stands for conjunction of primitive constraints.
- $\mathcal{K}$ in the solved form, example:
  - $x \doteq f(a, X) \wedge Y \doteq (a, f(b), X) \wedge X$ in $f(a)^* \cdot b$.
- $\mathcal{K}$ not in the solved form, examples:
  1. $x \doteq f(a, X) \wedge (Y, a) \doteq (a, f(b), X) \wedge X$ in $f(a)^* \cdot b$.
  2. $x \doteq f(a, X) \wedge (Y, a, f(b)) \doteq (a, f(b), Y) \wedge X$ in $f(a)^* \cdot b$.
  3. $x \doteq f(a, X) \wedge X$ in $f(a)^* \cdot b \wedge X$ in a^*.

Constraints

- $\mathcal{K}$ stands for conjunction of primitive constraints.
- $\mathcal{K}$ in the solved form, example:
  - $x \doteq f(a, X) \wedge Y \doteq (a, f(b), X) \wedge X$ in $f(a)^* \cdot b$.
- $\mathcal{K}$ not in the solved form, examples:
  1. $x \doteq f(a, X) \wedge (Y, a) \doteq (a, f(b), X) \wedge X$ in $f(a)^* \cdot b$.
  2. $x \doteq f(a, X) \wedge (Y, a, f(b)) \doteq (a, f(b), Y) \wedge X$ in $f(a)^* \cdot b$.
  3. $x \doteq f(a, X) \wedge X$ in $f(a)^* \cdot b \wedge X$ in a^*.
  4. $x \doteq f(a, Y) \wedge Y \doteq (a, f(b), X) \wedge X$ in $f(a)^* \cdot b$.

Constraints

- $\mathcal{K}$ stands for conjunction of primitive constraints.
- $\mathcal{K}$ in the solved form, example:
  - $x \doteq f(a, X) \wedge Y \doteq (a, f(b), X) \wedge X$ in $f(a)^* \cdot b$.
- $\mathcal{K}$ not in the solved form, examples:
  1. $x \doteq f(a, X) \wedge (Y, a) \doteq (a, f(b), X) \wedge X$ in $f(a)^* \cdot b$.
  2. $x \doteq f(a, X) \wedge (Y, a, f(b)) \doteq (a, f(b), Y) \wedge X$ in $f(a)^* \cdot b$.
  3. $x \doteq f(a, X) \wedge X$ in $f(a)^* \cdot b \wedge X$ in a^*.
  4. $x \doteq f(a, Y) \wedge Y \doteq (a, f(b), X) \wedge X$ in $f(a)^* \cdot b$.
  5. $f(x, b) \doteq f(f(a, X), b) \wedge Y \doteq (a, f(b), X) \wedge X$ in $f(a)^* \cdot b$.

Constraints

- $\mathcal{K}$ stands for conjunction of primitive constraints.
- $\mathcal{K}$ in the solved form, example:
  - $x \doteq f(a, X) \wedge Y \doteq (a, f(b), X) \wedge X \text{ in } f(a)^* \cdot b$.
- $\mathcal{K}$ not in the solved form, examples:
  1. $x \doteq f(a, X) \wedge (Y, a) \doteq (a, f(b), X) \wedge X \text{ in } f(a)^* \cdot b$.
  2. $x \doteq f(a, X) \wedge (Y, a, f(b)) \doteq (a, f(b), Y) \wedge X \text{ in } f(a)^* \cdot b$.
  3. $x \doteq f(a, X) \wedge X \text{ in } f(a)^* \cdot b \wedge X \text{ in } a^*$.
  4. $x \doteq f(a, Y) \wedge Y \doteq (a, f(b), X) \wedge X \text{ in } f(a)^* \cdot b$.
  5. $f(x, b) \doteq f(f(a, X), b) \wedge Y \doteq (a, f(b), X) \wedge X \text{ in } f(a)^* \cdot b$.
- The constraints 1–2 are in the partially solved form.
- The constraint 3–5 are not even partially solved.

Constraints

- $\mathcal{K}$ stands for conjunction of primitive constraints.
- $\mathcal{K}$ in the solved form, example:
  - $x \doteq f(a, X) \wedge Y \doteq (a, f(b), X) \wedge X$ in $f(a)^* \cdot b$.
- $\mathcal{K}$ not in the solved form, examples:
  1. $x \doteq f(a, X) \wedge (Y, a) \doteq (a, f(b), X) \wedge X$ in $f(a)^* \cdot b$.
  2. $x \doteq f(a, X) \wedge (Y, a, f(b)) \doteq (a, f(b), Y) \wedge X$ in $f(a)^* \cdot b$.
  3. $x \doteq f(a, X) \wedge X$ in $f(a)^* \cdot b \wedge X$ in a^*.
  4. $x \doteq f(a, Y) \wedge Y \doteq (a, f(b), X) \wedge X$ in $f(a)^* \cdot b$.
  5. $f(x, b) \doteq f(f(a, X), b) \wedge Y \doteq (a, f(b), X) \wedge X$ in $f(a)^* \cdot b$.
- The constraints 1–2 are in the partially solved form.
- The constraint 3–5 are not even partially solved.
- Every solved constraint is partially solved, but not vice versa.
- true is solved, false is not partially solved.

Constraints

Notation:

- $\mathcal{C}$: A constraint in DNF $\mathcal{K}_1 \vee \cdots \vee \mathcal{K}_n$.
- $\mathfrak{I}$: An intended structure.

Theorem

If $\mathcal{C}$ is solved, then $\mathfrak{I} \models \exists \mathcal{C}$.

Constraint Solver

- A rule-based algorithm, denoted $solve$.
- Input: a constraint in DNF.
- Output: a constraint in DNF.

Properties:

Theorem

If $solve(\mathcal{C}_{\mathsf{in}}) = \mathcal{C}_{\mathsf{out}}$, *then*

- $\mathcal{C}_{\mathsf{out}}$ *is equivalent to* $\mathcal{C}_{\mathsf{in}}$*,*
- $\mathcal{C}_{\mathsf{out}}$ *is* false *or in partially solved form.*

Constraint Solver

Example

- Input to the solver:

$$f(X, F(Y), Z) \doteq f(a, x, f(X)) \wedge X \text{ in } a(b^*) \cdot a(b^*)^*$$

- Output:

$$\begin{aligned} & X \doteq a \wedge x \doteq F(Y) \wedge Z \doteq f(a) \\ \vee\ & X \doteq (a, x) \wedge F \doteq f \wedge Y \doteq (a, x) \wedge Z \doteq \epsilon \wedge x \text{ in } a(b^*)^* \end{aligned}$$

- The output is in the solved form.

Constraint Solver

Example

- Input to the solver:

$$f(g(X), f(a, X)) \doteq f(f(Y, a), f(X, a))$$

- Output:

$$X \doteq (Y, a) \wedge (a, Y) \doteq (Y, a)$$

- The output is in the partially solved form.

Special Fragments

- What kind of constraints are reduced by $solve$ either to false or to a solved form?
- We identified two such fragments:
  - well-moded fragment
  - KIF fragment

Well-Moded Constraints

- A conjunction of primitive constraints $\mathcal{K} = \pi_1 \wedge \cdots \wedge \pi_n$ is well-moded, if for each $1 \leq i \leq n$,

Well-Moded Constraints

- A conjunction of primitive constraints $\mathcal{K} = \pi_1 \wedge \cdots \wedge \pi_n$ is well-moded, if for each $1 \leq i \leq n$,
  - if π_i is $t_1 \doteq t_2$, then either all variables of t_1 or all variables of t_2 occur in $\pi_1 \wedge \cdots \wedge \pi_{i-1}$,

Well-Moded Constraints

- A conjunction of primitive constraints $\mathcal{K} = \pi_1 \wedge \cdots \wedge \pi_n$ is well-moded, if for each $1 \leq i \leq n$,
  - if π_i is $t_1 \doteq t_2$, then either all variables of t_1 or all variables of t_2 occur in $\pi_1 \wedge \cdots \wedge \pi_{i-1}$,
  - if π_i is H in R, then all variables of H occur in $\pi_1 \wedge \cdots \wedge \pi_{i-1}$.

Well-Moded Constraints

- A conjunction of primitive constraints $\mathcal{K} = \pi_1 \wedge \cdots \wedge \pi_n$ is well-moded, if for each $1 \leq i \leq n$,
  - if π_i is $t_1 \doteq t_2$, then either all variables of t_1 or all variables of t_2 occur in $\pi_1 \wedge \cdots \wedge \pi_{i-1}$,
  - if π_i is H in R, then all variables of H occur in $\pi_1 \wedge \cdots \wedge \pi_{i-1}$.
- A constraint $\mathcal{C} = \mathcal{K}_1 \vee \cdots \vee \mathcal{K}_n$ is well-moded, if each $\mathcal{K}_i$ is well-moded.

Well-Moded Constraints

- A conjunction of primitive constraints $\mathcal{K} = \pi_1 \wedge \cdots \wedge \pi_n$ is well-moded, if for each $1 \leq i \leq n$,
  - if π_i is $t_1 \doteq t_2$, then either all variables of t_1 or all variables of t_2 occur in $\pi_1 \wedge \cdots \wedge \pi_{i-1}$,
  - if π_i is H in R, then all variables of H occur in $\pi_1 \wedge \cdots \wedge \pi_{i-1}$.
- A constraint $\mathcal{C} = \mathcal{K}_1 \vee \cdots \vee \mathcal{K}_n$ is well-moded, if each $\mathcal{K}_i$ is well-moded.
- $F_1(X, y, Z) \doteq f(a, b) \wedge F_1(a, Z) \doteq F_2(x, Y, X) \wedge Y$ in a^* is a well-moded constraint.

Well-Moded Constraints

- A conjunction of primitive constraints $\mathcal{K} = \pi_1 \wedge \cdots \wedge \pi_n$ is well-moded, if for each $1 \leq i \leq n$,
  - if π_i is $t_1 \doteq t_2$, then either all variables of t_1 or all variables of t_2 occur in $\pi_1 \wedge \cdots \wedge \pi_{i-1}$,
  - if π_i is H in R, then all variables of H occur in $\pi_1 \wedge \cdots \wedge \pi_{i-1}$.
- A constraint $\mathcal{C} = \mathcal{K}_1 \vee \cdots \vee \mathcal{K}_n$ is well-moded, if each $\mathcal{K}_i$ is well-moded.
- $F_1(X, y, Z) \doteq f(a, b) \wedge F_1(a, Z) \doteq F_2(x, Y, X) \wedge Y$ in a^* is a well-moded constraint.
- $F_1(X, y, Z) \doteq f(a, X) \wedge F_1(a, Z) \doteq F_2(x, Y, X) \wedge Y$ in a^* is not a well-moded constraint.

KIF Constraints

- A constraint $\mathcal{C} = \mathcal{K}_1 \vee \cdots \vee \mathcal{K}_n$ is in KIF form, if all hedge variables appear in the last positions.

KIF Constraints

- A constraint $\mathcal{C} = \mathcal{K}_1 \vee \cdots \vee \mathcal{K}_n$ is in KIF form, if all hedge variables appear in the last positions.
- $(x, y, Y) \doteq (f(X), b, X) \wedge F(a, Z) \doteq F(x, g(Y), g(a, b, X)) \wedge (a, Y)$ in a^* is a KIF constraint.

KIF Constraints

- A constraint $\mathcal{C} = \mathcal{K}_1 \vee \cdots \vee \mathcal{K}_n$ is in KIF form, if all hedge variables appear in the last positions.
- $(x, y, Y) \doteq (f(X), b, X) \wedge F(a, Z) \doteq F(x, g(Y), g(a, b, X)) \wedge (a, Y)$ in a^* is a KIF constraint.
- $(x, Y, y) \doteq (f(X), b, X) \wedge F(a, Z) \doteq F(x, g(Y), g(a, b, X)) \wedge (a, Y)$ in a^* is not a KIF constraint.

KIF Constraints

- A constraint $\mathcal{C} = \mathcal{K}_1 \vee \cdots \vee \mathcal{K}_n$ is in KIF form, if all hedge variables appear in the last positions.
- $(x, y, Y) \doteq (f(X), b, X) \wedge F(a, Z) \doteq F(x, g(Y), g(a, b, X)) \wedge (a, Y)$ in a^* is a KIF constraint.
- $(x, Y, y) \doteq (f(X), b, X) \wedge F(a, Z) \doteq F(x, g(Y), g(a, b, X)) \wedge (a, Y)$ in a^* is not a KIF constraint.
- $(x, y, Y) \doteq (f(X), b, X) \wedge F(a, Z) \doteq F(x, g(Y), g(a, b, X, c)) \wedge (a, Y)$ in a^* is not a KIF constraint either.

Well-Moded and KIF Constraints

Lemma

Let $\mathcal{C}$ be a well-moded or a KIF constraint and $solve(\mathcal{C}) = \mathcal{C}'$, where $\mathcal{C}' \neq$ false. Then $\mathcal{C}'$ is solved.

Relating to Programs

- Can we characterize programs that give rise well-moded or KIF constraints during derivations?

Relating to Programs

- Can we characterize programs that give rise well-moded or KIF constraints during derivations?
- Yes.

Relating to Programs

- Can we characterize programs that give rise well-moded or KIF constraints during derivations?
- Yes.
- Well-moded programs, KIF programs.

Relating to Programs

- Can we characterize programs that give rise well-moded or KIF constraints during derivations?
- Yes.
- Well-moded programs, KIF programs.
- KIF programs are easy: Just require that all occurrences of hedge variables happen in the last argument positions in subterms.

Relating to Programs

- Can we characterize programs that give rise well-moded or KIF constraints during derivations?
- Yes.
- Well-moded programs, KIF programs.
- KIF programs are easy: Just require that all occurrences of hedge variables happen in the last argument positions in subterms.
- Well-moded programs need a bit more involved definition.

Well-Moded Program

Example (Rewriting)

$$rewrite(x, y) \leftarrow rule(x, y).$$
$$rewrite(F(X, x, Y), F(X, y, Y)) \leftarrow rewrite(x, y).$$

$$rule(f(X), f(b, X, b)) \leftarrow X \text{ in } a^*.$$

Summary

- CLP(H) programs explore benefits of different kinds of variables and unranked symbols.
- The programs are short, yet quite clear and intuitive.
- CLP(H) generalizes languages such as, e.g., CLP(Flex) (Coelho and Florido, 2004), CLP($\mathcal{S}$) (Rajasekar, 1994), CLP(Σ^*) (Walinsky, 1989).
- Semantics of CLP(H) has been studied.
- A constraint solver, which computes partial solutions, has been developed.
- Two fragments (well-moded, KIF), which can be solved completely, have been identified.