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

\[ f(g, f(X), g(F(a), y)) \quad \{ X \mapsto (g(a), y), y \mapsto f(a), F \mapsto g \} \]
Three kinds of variables give flexibility of term traversal.

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

It helps to write short, yet quite clear and intuitive code.

Example (Rewriting)

\[
\begin{align*}
\text{rewrite}(x, y) &\leftarrow \text{rule}(x, y). \\
\text{rewrite}(F(X, x, Y), F(X, y, Y)) &\leftarrow \text{rewrite}(x, y).
\end{align*}
\]

\[
\begin{align*}
\text{rule}(x, y) &\leftarrow \ldots \\
\ldots
\end{align*}
\]
CLP(H) Programs

- Hedges may be constrained with regular hedge languages.
Hedges may be constrained with regular hedge languages.

Example (Rewriting)

\[
\text{rewrite}(x, y) \leftarrow \text{rule}(x, y).
\]
\[
\text{rewrite}(F(X, x, Y), F(X, y, Y)) \leftarrow \text{rewrite}(x, y).
\]
\[
\text{rule}(f(X), f(b, X, b)) \leftarrow X \text{ in } a^*.
\]
\[\ldots\]
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, \( \equiv \), 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:

\[ H ::= h_1, \ldots, h_n, \quad n \geq 0. \]

Notation:

\[ x: \text{ term variable} \quad f: \text{ function symbol} \]
\[ F: \text{ function variable} \quad X: \text{ hedge variable} \]
Let’s Get a Bit Formal

- Regular hedge expressions:

\[
R ::= \text{eps} \quad (\text{empty hedge expression}) \\
| R \cdot R \quad (\text{concatenation}) \\
| R + R \quad (\text{choice}) \\
| R^* \quad (\text{repetition}) \\
| f(R) \quad (\text{function application})
\]
Let’s Get a Bit Formal

- Regular hedge expressions:

  \[ R ::= \begin{array}{ll}
  \text{eps} & \text{(empty hedge expression)} \\
  | R \cdot R & \text{(concatenation)} \\
  | R + R & \text{(choice)} \\
  | R^* & \text{(repetition)} \\
  | f(R) & \text{(function application)}
  \end{array} \]

Example

- \[ f((a(\text{eps}) + b(\text{eps}))^*) \cdot c(\text{eps})^* \] is a regular hedge expression.
- For simplicity, it is written as \[ f((a + b)^*) \cdot c^* \].
More Notions

Primitive constraints:
- Equalities: $t_1 \equiv t_2$.
- Membership atoms: $H$ in R.
More Notions

Primitive constraints:

▶ Equalities: \( t_1 \equiv t_2 \).
▶ Membership atoms: \( H \in R \).

Example

▶ Equational primitive constraints:
  ▶ \( f(X, a) = f(a, X) \).
  ▶ \( f(X, F(Y), Z) \equiv f(a, x, f(X)) \).
More Notions

Primitive constraints:

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

Example

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

- $\text{rewrite}(F(X, x, Y), F(X, y, Y))$ 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\[\forall (L_1 \land \cdots \land L_n \rightarrow A),\] 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 \( \exists (L_1 \land \cdots \land L_n), \) \( n \geq 0, \)
  written as
  \[ \leftarrow L_1, \ldots, L_n. \]
CLP(H) Programs and Goals: Examples

- Program for removing duplicate arguments from a term:

  \[
  \text{remove_duplicates}(F(X, x, Y, x, Z), y) \leftarrow \\
  \text{remove_duplicates}(F(X, x, Y, Z), y). \\
  \text{remove_duplicates}(x, x).
  \]

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

  \[
  \leftarrow \text{remove_duplicates}(f(a, g(b), g(b), a, c), y).
  \]
A program that implements the rewriting mechanism, together with a rule to perform rewritings of the form
\[ f \rightarrow f(b, b), \ f(a) \rightarrow f(b, a, b), \ f(a, a) \rightarrow f(b, a, a, b), \text{ etc.} \]

\[
\begin{align*}
\text{rewrite}(x, y) & \leftarrow \text{rule}(x, y). \\
\text{rewrite}(F(X, x, Y), F(X, y, Y)) & \leftarrow \text{rewrite}(x, y).
\end{align*}
\]

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

Goal: Find a term that rewrites to \( f(a, f(b, f(b, a, a, b))) \): \( \leftarrow \text{rewrite}(x, f(a, f(b, f(b, a, a, b)))) \).
Declarative Semantics

- A **structure** for our language: $\mathcal{G} := \langle D, I \rangle$.

- $D$: 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 $\mathcal{G}$: 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 $\mathcal{S} = \langle D, I \rangle$ and a variable assignment $\sigma$.

- **Terms** are interpreted as elements of $D$:

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

- **Hedges** are interpreted as elements of $D^*$:

  $$ [(H_1, \ldots, H_n)]_{\mathcal{S},\sigma} := ([H_1]_{\mathcal{S},\sigma}, \ldots, [H_n]_{\mathcal{S},\sigma}). $$
Declarative Semantics

Interpretation of syntactic categories with respect to a structure $\mathcal{G} = \langle D, I \rangle$ and a variable assignment $\sigma$.

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

$$
\begin{align*}
[\text{eps}]_\mathcal{G} & := \{ \epsilon \}, \\
[R_1 \cdot R_2]_\mathcal{G} & := \{ (H_1, H_2) \mid H_1 \in [R_1]_\mathcal{G}, H_2 \in [R_2]_\mathcal{G} \}, \\
[R_1 + R_2]_\mathcal{G} & := [R_1]_\mathcal{G} \cup [R_2]_\mathcal{G}, \\
[R^*]_\mathcal{G} & := [R]^*_\mathcal{G}, \\
[f(R)]_\mathcal{G} & := \{ I(f)(H) \mid H \in [R]_\mathcal{G} \},
\end{align*}
$$
Declarative Semantics

Interpretation of syntactic categories with respect to a structure \( \mathcal{G} = \langle D, I \rangle \) and a variable assignment \( \sigma \).

- Primitive equational constraints are interpreted as equality:
  \[
  \mathcal{G} \models_\sigma t_1 = t_2 \iff [t_1]_{\mathcal{G},\sigma} = [t_2]_{\mathcal{G},\sigma}.
  \]

- Primitive membership constraints are interpreted as set membership:
  \[
  \mathcal{G} \models_\sigma H \text{ in } R \iff [H]_{\mathcal{G},\sigma} \in [R]_{\mathcal{G}}.
  \]

- Other formulas are interpreted in the standard way.
Intended structure: $\mathcal{J} = \langle D, I \rangle$, where

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

Intended structure: \( \mathcal{I} = \langle D, I \rangle \), 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:

- $\mathcal{G} \models A$: $\mathcal{G}$ 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 $\text{lm}(P)$.
2. For every program $P$ and goal $G$, $P \models G$ iff $\text{lm}(P) \models G$. 
Constraints

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

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

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

Constraints

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

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

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

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

- $\mathcal{K}$ stands for conjunction of primitive constraints.
- $\mathcal{K}$ in the solved form, example:
  - $x \equiv f(a, X) \land Y \equiv (a, f(b), X) \land X \text{ in } f(a)^* \cdot b$.
- $\mathcal{K}$ not in the solved form, examples:
  1. $x \equiv f(a, X) \land (Y, a) \equiv (a, f(b), X) \land X \text{ in } f(a)^* \cdot b$.
  2. $x \equiv f(a, X) \land (Y, a, f(b)) \equiv (a, f(b), Y) \land X \text{ in } f(a)^* \cdot b$.
  3. $x \equiv f(a, X) \land X \text{ in } f(a)^* \cdot b \land X \text{ in } a^*$.
  4. $x \equiv f(a, Y) \land Y \equiv (a, f(b), X) \land X \text{ in } f(a)^* \cdot b$.
  5. $f(x, b) \equiv f(f(a, X), b) \land Y \equiv (a, f(b), X) \land X \text{ in } f(a)^* \cdot b$.
- 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:
  - $x \doteq f(a, X) \land Y \doteq (a, f(b), X) \land X$ in $f(a)^* \cdot b$.
- $\mathcal{K}$ not in the solved form, examples:
  1. $x \doteq f(a, X) \land (Y, a) \doteq (a, f(b), X) \land X$ in $f(a)^* \cdot b$.
  2. $x \doteq f(a, X) \land (Y, a, f(b)) \doteq (a, f(b), Y) \land X$ in $f(a)^* \cdot b$.
  3. $x \doteq f(a, X) \land X$ in $f(a)^* \cdot b \land X$ in $a^*$.
  4. $x \doteq f(a, Y) \land Y \doteq (a, f(b), X) \land X$ in $f(a)^* \cdot b$.
  5. $f(x, b) \doteq f(f(a, X), b) \land Y \doteq (a, f(b), X) \land X$ 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 $\mathcal{K}_1 \lor \cdots \lor \mathcal{K}_n$.
- $\mathcal{I}$: An intended structure.

**Theorem**

*If $\mathcal{C}$ is solved, then $\mathcal{I} \models \exists \mathcal{C}$.*
Constraint Solver

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

Properties:

**Theorem**

If $solve(C_{in}) = C_{out}$, then

- $C_{out}$ is equivalent to $C_{in}$,
- $C_{out}$ is false or in partially solved form.
**Constraint Solver**

**Example**

- **Input to the solver:**

\[
f(X, F(Y), Z) \equiv f(a, x, f(X)) \land X \text{ in } a(b^*) \cdot a(b^*)^*
\]

- **Output:**

\[
X \equiv a \land x \equiv F(Y) \land Z \equiv f(a) \\
\lor X \equiv (a, x) \land F \equiv f \land Y \equiv (a, x) \land Z \equiv \epsilon \land x \text{ in } a(b^*)^*
\]

- **The output is in the solved form.**
Example

- Input to the solver:

\[ f(g(X), f(a, X)) = f(f(Y, a), f(X, a)) \]

- Output:

\[ X = (Y, a) \land (a, Y) \]

- 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 $\mathcal{K} = \pi_1 \land \cdots \land \pi_n$ is well-moded, if for each $1 \leq i \leq n$, if $\pi_i$ is $t_1 = t_2$, then either all variables of $t_1$ or all variables of $t_2$ occur in $\pi_1 \land \cdots \land \pi_{i-1}$. If $\pi_i$ is $H \in R$, then all variables of $H$ occur in $\pi_1 \land \cdots \land \pi_{i-1}$.

A constraint $C = K_1 \lor \cdots \lor K_n$ is well-moded, if each $K_i$ is well-moded.

$F_1(X,y,Z) = f(a,b) \land F_1(a,Z) = F_2(x,Y,X) \land Y$ in $a^*$ is a well-moded constraint.

$F_1(X,y,Z) = f(a,X) \land F_1(a,Z) = F_2(x,Y,X) \land Y$ in $a^*$ is not a well-moded constraint.
Well-Moded Constraints

- A conjunction of primitive constraints $\mathcal{K} = \pi_1 \land \cdots \land \pi_n$ is well-moded, if for each $1 \leq i \leq n$,
  - if $\pi_i$ is $t_1 \vdash t_2$, then either all variables of $t_1$ or all variables of $t_2$ occur in $\pi_1 \land \cdots \land \pi_{i-1}$,
A conjunction of primitive constraints \( \mathcal{K} = \pi_1 \land \cdots \land \pi_n \) is well-moded, if for each \( 1 \leq i \leq n \),
- if \( \pi_i \) is \( t_1 \equiv t_2 \), then either all variables of \( t_1 \) or all variables of \( t_2 \) occur in \( \pi_1 \land \cdots \land \pi_{i-1} \),
- if \( \pi_i \) is \( H \) in \( R \), then all variables of \( H \) occur in \( \pi_1 \land \cdots \land \pi_{i-1} \).
Well-Moded Constraints

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

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

- A conjunction of primitive constraints $K = \pi_1 \land \cdots \land \pi_n$ is well-moded, if for each $1 \leq i \leq n$,
  - if $\pi_i$ is $t_1 \equiv t_2$, then either all variables of $t_1$ or all variables of $t_2$ occur in $\pi_1 \land \cdots \land \pi_{i-1}$,
  - if $\pi_i$ is $H$ in $R$, then all variables of $H$ occur in $\pi_1 \land \cdots \land \pi_{i-1}$.
- A constraint $C = K_1 \lor \cdots \lor K_n$ is well-moded, if each $K_i$ is well-moded.
  - $F_1(X, y, Z) \equiv f(a, b) \land F_1(a, Z) \equiv F_2(x, Y, X) \land Y$ in $a^*$ is a well-moded constraint.
  - $F_1(X, y, Z) \equiv f(a, X) \land F_1(a, Z) \equiv F_2(x, Y, X) \land Y$ in $a^*$ is not a well-moded constraint.
KIF Constraints

- A constraint $C = K_1 \lor \cdots \lor K_n$ is in KIF form, if all hedge variables appear in the last positions.
KIF Constraints

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

- $(x, y, Y) \vDash (f(X), b, X) \land F(a, Z) \vDash F(x, g(Y), g(a, b, X)) \land (a, Y)$ in $a^*$ is a KIF constraint.
KIF Constraints

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

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

▶ $(x, y, Y) \vdash (f(X), b, X) \land F(a, Z) \vdash F(x, g(Y), g(a, b, X)) \land (a, Y)$ in $a^*$ is a KIF constraint.

▶ $(x, Y, y) \vdash (f(X), b, X) \land F(a, Z) \vdash F(x, g(Y), g(a, b, X)) \land (a, Y)$ in $a^*$ is not a KIF constraint.

▶ $(x, y, Y) \vdash (f(X), b, X) \land F(a, Z) \vdash F(x, g(Y), g(a, b, X, c)) \land (a, Y)$ in $a^*$ is not a KIF constraint either.
Well-Moded and KIF Constraints

Lemma

Let $C$ be a well-moded or a KIF constraint and $\text{solve}(C) = C'$, where $C' \neq \text{false}$. Then $C'$ is solved.
Can we characterize programs that give rise well-moded or KIF constraints during derivations?
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.
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.
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.
Example (Rewriting)

\[
\text{rewrite}(x, y) \leftarrow \text{rule}(x, y).
\]

\[
\text{rewrite}(F(X, x, Y), F(X, y, Y)) \leftarrow \text{rewrite}(x, y).
\]

\[
\text{rule}(f(X), f(b, X, b)) \leftarrow X \text{ in } a^*.
\]
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(S) (Rajasekar, 1994), CLP($\Sigma^*$) (Walinsky, 1989).
- Semantics of CLP(H) has been studied.
- A constraint solver, which computes partial solutions, has been developed.
- Two fragments (well-modeled, KIF), which can be solved completely, have been identified.