

# Coinductive Functional Logic Programming

Ronald de Haan

TU Dresden

February 18, 2011

# Outline

## 1 Background

- Coinductive Logic Programming
- Functional Logic Programming

## 2 Future work

- Coinductive Functional Logic Programming

# Outline

## 1 Background

- Coinductive Logic Programming
- Functional Logic Programming

## 2 Future work

- Coinductive Functional Logic Programming

# Intuitions behind CLP

- Logic programming works with inductively defined (finite) terms
- Coinductive logic programming computes with infinite (cyclic) terms
- The idea: use coinduction to finitely reason about infinite things

# Machinery

- A coinductive logic program is a definite program with maximal co-Herbrand model declarative semantics.
  - ▶ greatest fixed point interpretation
- Operational semantics: co-SLD-resolution
  - ▶ regular SLD-resolution steps
  - ▶ “coinductive hypothesis” step:
    - ★ if a goal  $Q$  is called, and  $Q$  unifies with a call made earlier, then  $Q$  succeeds
    - ★ you can think of this as a circularity check
- Operational semantics is sound and complete w.r.t. the declarative semantics [1]

## Example

- Coinductive logic program

$$\mathcal{P} = \{ \begin{array}{l} \text{bit}(0). \\ \text{bit}(1). \\ \text{bitstream}([H|T]) \text{ :- bit}(H), \text{bitstream}(T). \end{array} \}$$

- The following query succeeds

?- X = [1,0,1,1|X], bitstream(X).

# Advantages

- Declarative programming with infinite terms
  - ▶ Graphs with cycles
  - ▶ Automata over infinite strings
  - ▶ Bisimilarity
  - ▶ Infinite processes
- Different approaches to other areas possible
  - ▶ e.g., predicate answer set programming with a top-down approach [2]

# Outline

## 1 Background

- Coinductive Logic Programming
- Functional Logic Programming

## 2 Future work

- Coinductive Functional Logic Programming

# Intuitions behind FLP

- Combining functional and logic programming
  - ▶ efficient execution principles of functional programming
  - ▶ flexibility of logic programming
- In the following, we look at (the principles behind) the language Curry in particular

# Formal machinery

- Needed narrowing [3]
  - ▶ narrowing is “rewriting with binding”
  - ▶ every step consists of binding and reducing
  - ▶ this happens in a way that is optimal
    - ★ no unnecessary steps,
    - ★ shortest derivations,
    - ★ minimal set of computed solutions
  - ▶ used narrowing strategy can be computed from a program
- Sound and complete for a large class of rewrite systems

# Example

- Rewrite rules:

$$0 \leq x \rightarrow \text{true}$$

$$(S\ x) \leq 0 \rightarrow \text{false}$$

$$(S\ x) \leq (S\ y) \rightarrow x \leq y$$

- One possible needed narrowing derivation (not the only one!):

$$(S\ x) \leq (S\ (S\ y)) \rightsquigarrow_{r_{3,\epsilon}} x \leq (S\ y) \rightsquigarrow_{r_{1,\{x \mapsto 0\}}} \text{true}$$

# Advantages

- (More) efficient execution of programs with free variables
- Can be (fairly) easily extended with
  - ▶ conditional rewrite rules
  - ▶ constraints
  - ▶ concurrency
- Integration of declarative paradigms

# Outline

## 1 Background

- Coinductive Logic Programming
- Functional Logic Programming

## 2 Future work

- Coinductive Functional Logic Programming

# Intuitions

- The (current) idea: extend the needed narrowing approach with a circularity check, similar to the extension of LP to coinductive LP
- This would result in a bisimilarity relation, instead of (or next to) an equality relation





# Formal machinery

- Proof theoretic framework for circular coinduction [4]
- ...
- Suggestions?

# Questions or suggestions?

- All suggestions are most welcome
- Don't hesitate to contact me  
([Ronald.de\\_Haan@mailbox.tu-dresden.de](mailto:Ronald.de_Haan@mailbox.tu-dresden.de))

# References

-  L. E. Simon, *Extending Logic Programming with Coinduction*. PhD thesis, University of Texas at Dallas, 2006.
-  R. Min, *Predicate Answer Set Programming with Coinduction*. PhD thesis, University of Texas at Dallas, 2009.
-  S. Antoy, R. Echahed, and M. Hanus, “A needed narrowing strategy,” in *POPL*, pp. 268–279, 1994.
-  G. Roşu and D. Lucanu, “Circular coinduction: a proof theoretical foundation,” in *Proceedings of the 3rd international conference on Algebra and coalgebra in computer science, CALCO'09*, (Berlin, Heidelberg), pp. 127–144, Springer-Verlag, 2009.