Isomorphisms for context-free types

Wouter Swierstra

April 19, 2006

The University of Nottingham

What are context-free types?

Context-free types over an index set I are built from:

$0, \sigma + \tau$	coproducts
$1, \sigma \times \tau$	products
$i \in I$	parameters
$X, Y, \ldots$	recursive variables
$\mu X.\sigma$	least fixed point

For instance:

- Lists: $\mu X.1 + A \times X$
- Binary trees: $\mu X.1 + X \times A \times X$

Context-free types resemble context-free grammars.

What is an isomorphism?

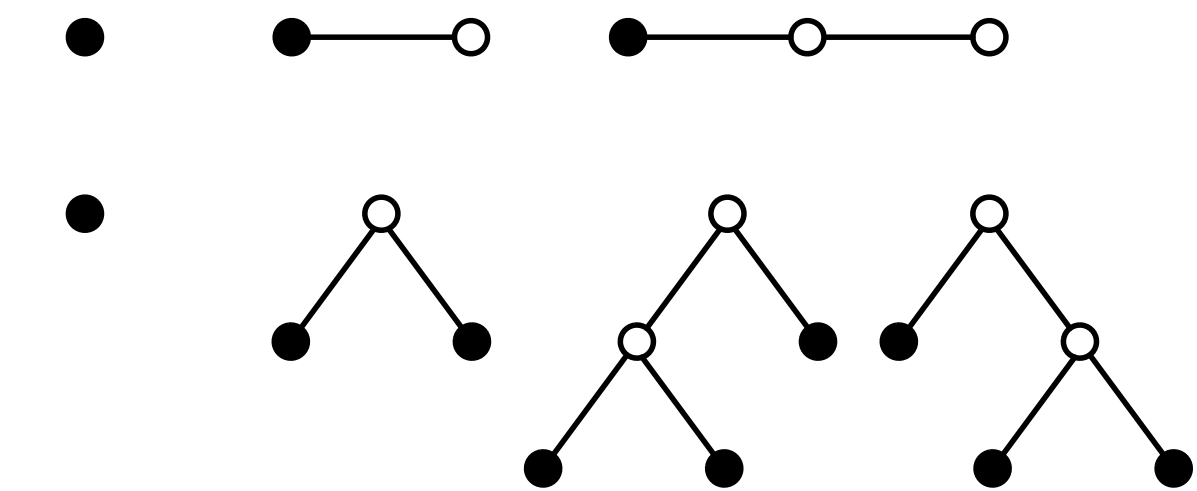
- We can interpret a context-free type σ as a functor $[\![\sigma]\!] : \mathbf{C}^I \to \mathbf{C}$.
- Two types σ and τ are isomorphic iff $[\![\sigma]\!]$ and $[\![\tau]\!]$ are naturally isomorphic.
- Is there structured fashion to decide whether or not two context-free types are isomorphic?

When are two types different?

- We can show two data types are distinct by counting the number of inhabitants.
- Are the following types isomorphic?
  - Lists: $\mu X.1 + A \times X$
  - Binary trees: $\mu X.1 + X \times A \times X$

When are two types different?

- We can show two data types are distinct by counting the number of inhabitants.
- Are the following types isomorphic?
  - Lists: $\mu X.1 + A \times X$
  - Binary trees: $\mu X.1 + X \times A \times X$



$n = 0$ $n = 1$ $n = 2$

How can we count the number of inhabitants of a type?

- Parsers tell us if a string is in a language or not:

$$I^* \to 2$$

- Parser combinators are more general and compose nicely:

$$I^* \to \mathcal{P}_{\mathsf{fin}}(I^*)$$

- Can we port this technology?

Monadic parser combinators

- If A is a set, $\mathcal{P}_{\text{fin}}(A)$ and A^* are monoids.
- $\mathcal{P}_{\text{fin}}(-)$ and $-^*$ are both monads.
- We can define parser combinators using only these properties.
- We can count the number of inhabitants of a type by shifting to multisets:

$$\mathcal{M}(I) \to \mathbf{N}$$

- By writing combinators of the following type:

$$\mathcal{M}(I) \to \mathcal{M}(\mathcal{M}(I))$$

- We change the underlying structure, but use the same idea.

Powerseries

- The multiset parsers give us a new interpretation of our types.
- We consider a type σ over a singleton index set I as:

$$\sum_{n \in \mathbf{N}} a_n \times X^n$$

  where a_n is the result of running the σ parser on n.
- **Lemma** There is a full and faithful functor taking powerseries to functors $\mathbf{C}^I \to \mathbf{C}$.
- **Lemma** Two types are isomorphic iff their powerseries are equal.

Powerseries

- The multiset parsers give us a new interpretation of our types.
- We consider a type σ over a singleton index set I as:

$$\sum_{n \in \mathbf{N}} a_n \times X^n$$

  where a_n is the result of running the σ parser on n.
- **Lemma** There is a full and faithful functor taking powerseries to functors $\mathbf{C}^I \to \mathbf{C}$.
- **Lemma** Two types are isomorphic iff their powerseries are equal.

Types up to isomorphism are powerseries.

Conclusions

- We have a semi-algorithm for deciding whether or not two types are isomorphic.
- Is the problem decidable?
- Is there a subset of types for which isomorphism is decidable?