Fixed Point Calculus

Roland Backhouse
December 10, 2002
Overview

- Why a calculus?
- Equational Laws
- Application
Specification ≠ Implementation

Suppose Prolog is being used to model family relations. Suppose \texttt{parent(X,Y)} represents the relationship \texttt{X} is a parent of \texttt{Y} and suppose \texttt{ancestor(X,Y)} is the transitive closure of the parent relation. Then

\[
\texttt{ancestor(X,Y) } \Leftarrow \texttt{parent(X,Y)}
\]

and

\[
\texttt{ancestor(X,Y) } \Leftarrow \exists \langle Z :: \texttt{ancestor(X,Z)} \land \texttt{ancestor(Z,Y)} \rangle .
\]

However,

\[
\texttt{ancestor(X,Y) } : - \texttt{parent(X,Y)} .
\]
\[
\texttt{ancestor(X,Y) } : - \texttt{ancestor(X,Z) , ancestor(Z,Y)} .
\]

is not a correct Prolog implementation.

\[
\texttt{ancestor(X,Y) } : - \texttt{parent(X,Y)} .
\]
\[
\texttt{ancestor(X,Y) } : - \texttt{parent(X,Z) , ancestor(Z,Y)} .
\]

is a correct implementation.
**Specification ≠ Implementation**

The grammar

\[
\langle \text{StatSeq} \rangle ::= \langle \text{Statement} \rangle \mid \langle \text{StatSeq} \rangle ; \langle \text{StatSeq} \rangle
\]

describes a sequence of statements separated by semicolons. But it is ambiguous and not amenable to top-down or bottom-up parsing.

The grammar

\[
\langle \text{StatSeq} \rangle ::= \langle \text{Statement} \rangle \langle \text{Rest} \rangle \\
\langle \text{Rest} \rangle ::= \varepsilon \mid ; \langle \text{Statement} \rangle \langle \text{Rest} \rangle
\]

is equivalent and amenable to parsing by recursive descent.

The grammar

\[
\langle \text{StatSeq} \rangle ::= \langle \text{Statement} \rangle \mid \langle \text{StatSeq} \rangle ; \langle \text{Statement} \rangle
\]

is also equivalent and preferable for bottom-up parsing.
Testing whether the empty word is generated by a grammar is easy. For example, given the grammar

\[
S ::= \varepsilon \mid aS
\]

we construct and solve the equation

\[
\varepsilon \in S = \varepsilon \in \{\varepsilon\} \lor (\varepsilon \in \{a\} \land \varepsilon \in S)
\]

But it is not the case that (eg)

\[
a \in S = a \in \{\varepsilon\} \lor (a \in \{a\} \land a \in S)
\]

(The least solution is \(a \in S = \text{false}\).)

The general membership test is a non-trivial problem!
Least Fixed Points

Recall the characterising properties of least fixed points:

*computation rule*

\[ \mu f = f.\mu f \]

*induction rule*: for all \( x \in A \),

\[ \mu f \leq x \iff f.x \leq x . \]

The induction rule is undesirable because it leads to proofs by mutual inclusion (i.e. the consideration of two separate cases).
Closure Rules

In any Kleene algebra

\[ a^* = \langle \mu x :: 1 + x \cdot a \rangle = \langle \mu x :: 1 + a \cdot x \rangle = \langle \mu x :: 1 + a + x \cdot x \rangle \]

\[ a^+ = \langle \mu x :: a + x \cdot a \rangle = \langle \mu x :: a + a \cdot x \rangle = \langle \mu x :: a + x \cdot x \rangle \]
Basic Rules

The *rolling rule*:

\[ \mu(f \circ g) = f.\mu(g \circ f) \ . \] (1)

The *square rule*:

\[ \mu f = \mu(f^2) \ . \] (2)

The *diagonal rule*:

\[ \langle \mu x :: x \oplus x \rangle = \langle \mu x :: \langle \mu y :: x \oplus y \rangle \rangle \ . \] (3)
Examples

\[ \langle \mu X :: a \cdot X^* \rangle = a^+ . \]

\[ \langle \mu X :: a + X \cdot b \cdot X \rangle = a \cdot (b \cdot a)^* . \]
Fusion

Many problems are expressed in the form

\[ \text{evaluate} \circ \text{generate} \]

where \text{generate} generates a (possibly infinite) candidate set of solutions, and \text{evaluate} selects a best solution.

Examples:

\[ \text{shortest} \circ \text{path} , \]

\[ (x \in) \circ L . \]

Solution method is to \textit{fuse} the generation and evaluation processes, eliminating the need to generate all candidate solutions.
Language Problems

\[ S ::= aSS | \varepsilon . \]

Is-empty

\[ S = \phi \equiv (\{a\} = \phi \lor S = \phi \lor S = \phi) \land \{\varepsilon\} = \phi . \]

Nullable

\[ \varepsilon \in S \equiv (\varepsilon \in \{a\} \land \varepsilon \in S \land \varepsilon \in S) \lor \varepsilon \in \{\varepsilon\} . \]

Shortest word length

\[ \#S = (\#a + \#S + \#S) \downarrow \#\varepsilon . \]

Non-Example

\[ aa \in S \neq (aa \in \{a\} \land aa \in S \land aa \in S) \lor aa \in \{\varepsilon\} . \]
Conditions for Fusion

Fusion is made possible when

- `evaluate` is an adjoint in a *Galois connection*,
- `generate` is expressed as a *fixed point*. 
Fusion Theorem

\[ F(\mu \preceq g) = \mu \sqsubseteq h \]

provided that

- \( F \) is a lower adjoint in a Galois connection of \( \sqsubseteq \) and \( \preceq \) (see brief summary of definition below)

- \( F \circ g = h \circ F \).

Galois Connection

\[ F.x \sqsubseteq y \equiv x \preceq G.y \]

\( F \) is called the lower adjoint and \( G \) the upper adjoint.
Shortest Word Problem

Given a language $L$ defined by a context-free grammar, determine the length of the shortest word in the language.

For concreteness, use the grammar

$$S ::= aS \mid SS \mid \varepsilon.$$ 

The language defined by this grammar is

$$\langle \mu X ::= \{a\} \cdot X \cup X \cdot X \cup \{\varepsilon\} \rangle.$$ 

Now, for arbitrary language $L$,

$$\#L = \langle \downarrow w : w \in L : \text{length}.w \rangle$$

and we are required to determine

$$\# \langle \mu X ::= \{a\} \cdot X \cup X \cdot X \cup \{\varepsilon\} \rangle.$$
Shortest Word Problem (Continued)

For arbitrary language $L$,

$$\#L = \langle \downarrow w : w \in L : \text{length.}w \rangle$$

and we are required to determine

$$\# \langle \mu X :: \{a\} \cdot X \cup X \cdot X \cup \{\varepsilon\} \rangle .$$

Because $\#$ is the infimum of the length function it is the lower adjoint in a Galois connection. Indeed,

$$\#L \geq k \equiv L \subseteq \Sigma^{\geq k}$$

where $\Sigma^{\geq k}$ is the set of all words (in the alphabet $\Sigma$) whose length is at least $k$.

So, by fusion, for all functions $f$ and $g$,

$$\#(\mu \subseteq f) = \mu \geq g \iff \# \circ f = g \circ \# .$$

Applying this to our example grammar, we fill in $f$ and calculate $g$ so that:

$$\# \circ \langle X :: \{a\} \cdot X \cup X \cdot X \cup \{\varepsilon\} \rangle = g \circ \# .$$
Shortest Word Problem (Continued)

\[ \# \circ \langle X :: \{a\} \cdot X \cup X \cdot X \cup \{\varepsilon\} \rangle = g \circ \# \]
\[ = \{ \text{definition of composition} \} \]
\[ \langle \forall X :: \#(\{a\} \cdot X \cup X \cdot X \cup \{\varepsilon\}) = g.(\#X) \rangle \]
\[ = \{ \# \text{ is a lower adjoint and so distributes over } \cup, \]
\[ \text{definition of } \# \} \]
\[ \langle \forall X :: \#(\{a\} \cdot X) \downarrow \#(X \cdot X) \downarrow \#\{\varepsilon\} = g.(\#X) \rangle \]
\[ = \{ \#(Y \cdot Z) = \#Y + \#Z, \#\{a\} = 1, \#\{\varepsilon\} = 0 \} \]
\[ (1 + \#X) \downarrow (\#X + \#X) \downarrow 0 = g.(\#X) \]
\[ \Leftarrow \{ \text{instantiation} \} \]
\[ \langle \forall k :: (1 + k) \downarrow (k + k) \downarrow 0 = g.k \rangle . \]

We conclude that
\[ \# \langle \mu X :: \{a\} \cdot X \cup X \cdot X \cup \{\varepsilon\} \rangle = \langle \mu k :: (1 + k) \downarrow (k + k) \downarrow 0 \rangle . \]
Language Recognition

**Problem:** For given word \( x \) and grammar \( G \), determine \( x \in L(G) \).
That is, implement

\[
(x \in) \circ L.
\]

Language \( L(G) \) is the least fixed point (with respect to the subset relation) of a monotonic function.

\( (x \in) \) is the lower adjoint in a Galois connection of languages (ordered by the subset relation) and booleans (ordered by implication).
(Recall,

\[
x \in S \Rightarrow b \quad \equiv \quad S \subseteq \text{if } b \rightarrow \Sigma^* \quad \Box \quad \neg b \rightarrow \Sigma^* - \{x\} \quad \text{fi}.
\]
Nullable Languages

**Problem:** For given grammar $G$, determine $\varepsilon \in L(G)$.

$$(\varepsilon \in) \circ L$$

**Solution:** Easily expressed as a fixed point computation.

Works because:

- The function $(x \in)$ is a lower adjoint in a Galois connection (for all $x$, but in particular for $x = \varepsilon$).
- For all languages $S$ and $T$,

$$\varepsilon \in S \cdot T \equiv \varepsilon \in S \land \varepsilon \in T.$$
Problem Generalisation

**Problem:** For given grammar $G$, determine whether all words in $L(G)$ have even length. I.e. implement

$$\text{alleven } \circ L .$$

The function *alleven* is a lower adjoint in a Galois connection. Specifically, for all languages $S$ and $T$,

$$\text{alleven}(S) \iff b \quad \equiv \quad S \subseteq \text{if } \neg b \rightarrow \Sigma^* \; \square b \rightarrow (\Sigma \cdot \Sigma)^* \quad \text{fi}$$

Nevertheless, fusion *doesn’t* work (directly) because

- there is no $\otimes$ such that, for all languages $S$ and $T$,

$$\text{alleven}(S \cdot T) \quad \equiv \quad \text{alleven}(S) \otimes \text{alleven}(T) .$$

**Solution:** Generalise by tupling: compute simultaneously *alleven* and *allodd*. 
General Context-Free Parsing

**Problem:** For given grammar $G$, determine $x \in L(G)$.

$$(x \in \circ L)$$

_Not_ (in general) expressible as a fixed point computation.

Fusion *fails* because: for all $x$, $x \neq \epsilon$, there is no $\otimes$ such that, for all languages $S$ and $T$,

$$x \in S \cdot T \equiv (x \in S) \otimes (x \in T).$$

**CYK:** Let $F(S)$ denote the relation $\langle i, j :: x[i..j] \in S \rangle$.

*Works* because:

- The function $F$ is a lower adjoint.
- For all languages $S$ and $T$,

$$F(S \cdot T) = F(S) \bullet F(T)$$

where $B \bullet C$ denotes the composition of relations $B$ and $C$. 