Insegnamento di

**Foundation of Computing**

Pagina del corso: [http://pages.di.unipi.it/montanari/FOC.html](http://pages.di.unipi.it/montanari/FOC.html)

Note di

**Logic Programming**

a cura di
Ugo Montanari
Dipartimento di Informatica
Università degli Studi di Pisa
ugo@di.unipi.it
A coalgebraic approach to unification semantics of logic programming

Ugo Montanari (UNIPI)

Giorgio Mossa (UNIPI) Roberto Bruni (UNIPI)

IFIP WG 1.3, Massa Marittima, Jan.14-16, 2020
Can we give a categorical account of the operational semantics of logic programs?
Inspiration

S-semantics

Non-ground atoms

Answer substitutions

S-semantics by Falaschi, Levi, Palamidessi and Martelli
Operational semantics
Logic program

logic signature:
$\Sigma$ operation symbols
$\Pi$ predicate symbols

atoms: $A \equiv P(t_1, \ldots, t_n)$
goals: $G \equiv A_1, \ldots, A_k$

$\mathcal{P}$ a set of Horn clauses  $H : \neg B$
Operational semantics

\[ P \models G \Rightarrow_\sigma F \]
Operational semantics

\[ H : - B \in \mathbb{P} \quad \sigma = \text{mgu}(A, \rho(H)) \]

\[ \mathbb{P} \models A \Rightarrow_{\sigma} \sigma(\rho(B)) \]

\(\rho(H)\) and \(\rho(B)\) have no variable in common with \(A\)
Operational semantics

\[ \mathbb{P} \models G \Rightarrow_\sigma F \]

\[ \mathbb{P} \models G, G' \Rightarrow_\sigma F, \sigma(G') \]
Canonical predicate

\{x_1, \ldots, x_n\} canonical set of variables

\[ P(t_1, \ldots, t_n) \text{ as } \sigma_t(P(x_1, \ldots, x_n)) \]

\( \sigma_t : \{x_1, \ldots, x_n\} \to T_\Sigma(X) \) with \( \sigma_t(x_i) = t_i \)

we write \( P \) for \( P(x_1, \ldots, x_n) \)
Definition 7 (Most general unifier II).

As before, the relation of being more general induces a preorder on the unifiers.

Remark 3.

Every atom $t$ is a unifier which is most general among all the possible unifiers.

Unification as a pushout

$$\text{mgu}(\alpha(P), \beta(Q))?$$

$$P = Q$$
Operational semantics

\[ \alpha(P) : - B \in \mathbb{P}, \quad (\alpha, \beta, \sigma_1, \sigma_2) \text{ is a pushout} \]

\[ \mathbb{P} \models \beta(P) \Rightarrow_{\sigma_2} \sigma_1(B) \]

no need of creating new variables
Coalgebras
Coalgebras

\[ F : \text{C} \to \text{C} \quad \langle X, \alpha : X \to F(X) \rangle \]

\[ \varrho_L : \text{Set} \to \text{Set} \quad \alpha : S \to \varrho(L \times S) \]

\[ \varrho_L(X) = \varrho(L \times X) \quad \alpha(s) = \{ (\ell, t) \mid s \xrightarrow{\ell} t \} \]
A coalgebraic approach to unification semantics of logic programming

Commute the diagram in Fig. 1c. This is equivalent to say that \((x_1, x_2)\) is a pushout square. So, unifiers are commutative squares in the form above and mgu's are those squares that are pushouts.

Using this characterization of unification via pushouts we can modify the Atomic goal rule of Table 1 with the following rule:

\[
(P) : \iff B_2 P (x_1, x_2)
\]

is a pushout

\[
P | = P (x_1, x_2)
\]

In this way the LTS is not changed, but there is no need of creating new variables.

2.2 LTSs as coalgebras.

Definition 8 (\(F\)-coalgebra).

Given a category \(C\) and an endofunctor \(F : C \to C\), a \(F\)-coalgebra, or \(F\) -coalgebra for short, is a pair \(h : (X, \alpha)\) with \(X\) an object of \(C\) and \(\alpha : X \to F(X)\). Given two coalgebras \(h : (X, \alpha)\) and \(h : (Y, \beta)\), a cohomomorphism \(h : (X, \alpha) \to (Y, \beta)\) is given by a morphism \(h : X \to Y\) such that the following diagram commutes.

\[
\begin{array}{ccc}
X & \xrightarrow{\alpha} & F(X) \\
\downarrow & & \downarrow F(h) \\
Y & \xrightarrow{\beta} & F(Y)
\end{array}
\]

Coalgebras provide an elegant way to encode different notions of dynamic systems in a categorical framework. As an example we can consider the case of LTSs. We recall that a labeled transition system is a triple \(h : (S, L, \to)\) where \(S\) is a set of states, \(L\) is a set of labels, and \(\to \subseteq S \times L \times S\) is a transition relation. Every LTS \(h : (S, L, \to)\) can be regarded as a coalgebra for the endofunctor \(P_L : \text{Set} \to \text{Set}\) defined on objects as \(P_L(X) = P(L \times X)\) where \(P\) is the powerset functor.

If \(h : (S, L, \to)\) is an LTS, the relation \(\to \subseteq S \times L \times S\) gives the function \(p : S \to P_L(S)\) that sends every \(s \in S\) into the set \(p(s) = \{l, t | s \to l t\} = \{l_1, l_2, \ldots, l_n\}\) to.

One of the most interesting thing about coalgebras is that they give an abstract semantics for dynamic systems in term of final objects: if \(h : (X, \alpha)\) is a coalgebra, we say that two states, i.e. two elements of \(X\), are bisimilar if they have the same image via the unique cohomomorphism \(h : (X, \alpha) \to (T, \xi)\) into a final coalgebra \(h : (T, \xi)\). This definition of bisimilarity generalizes the classical one for transition systems, meaning that two states are bisimilar in the classical sense if and only if they are bisimilar in the coalgebraic sense.

The advantage of coalgebras over classical LTS is that the states' space now can be an object of a generic category, not just a set. Hence coalgebras allow to work with states that have an additional structure. This justifies the name structured coalgebras for coalgebras over categories of structures, in particular we will be interested in coalgebras over categories of algebras.
Final coalgebra

\[ \langle T, \tau \rangle \quad \forall \langle X, \alpha \rangle. \ \exists ! t : \langle X, \alpha \rangle \rightarrow \langle T, \tau \rangle \]

\[ p \sim q \iff t(p) = t(q) \]
Algebraic structure of states
The theory of monoids

a sort $M$

$\cdot : M \times M \rightarrow M$

$\Box : \rightarrow M$

\[ x \cdot (y \cdot z) = (x \cdot y) \cdot z \]

\[ x \cdot \Box = x = \Box \cdot x \]
The theory of substitutions

a countable set of sorts \( \{n \mid n \in \mathbb{N}\} \)

\( \sigma : n \rightarrow m \) for every \( \sigma : \{x_1, \ldots, x_n\} \rightarrow T_\Sigma(\{x_1, \ldots, x_m\}) \)

\[ \tau(\sigma(x)) = \tau \circ \sigma(x) \]

\[ \text{id}_n(x) = x \]
The theory of substitutive monoids

Just a product theory!

a countable set of sorts \( \{ \underline{n} \mid n \in \mathbb{N} \} \)
\( \sigma : \underline{n} \to \underline{m} \) for every \( \sigma : \{x_1, \ldots, x_n\} \to T_\Sigma(\{x_1, \ldots, x_m\}) \)
\( \cdot_n : \underline{n} \times \underline{n} \to \underline{n} \)
\( \square_n : \to \underline{n} \)

\[
\tau(\sigma(x)) = \tau \circ \sigma(x) \\
x \cdot_n (y \cdot_n z) = (x \cdot_n y) \cdot_n z \\
\sigma(x \cdot_n y) = \sigma(x) \cdot_m \sigma(y) \\
\sigma(\square_n) = \square_m
\]

\( \text{id}_n(x) = x \)
(\( x \cdot_n \square_n = \square_n \cdot_n x = x \))
The theory of $\Pi$-substitutive monoids

Add to the theory of substitutive monoids

$P :\rightarrow n$ for every $P \in \Pi_n$
Initiability of the goal algebra

\[
A_1, \ldots, A_k
\]

\[
\sigma_1(P_1) \cdot \underbrace{n (\cdots \cdot n (\sigma_{k-1}(P_{k-1}) \cdot \underbrace{n \sigma_k(P_k)) \cdots))}_{\text{A syntax for states}}
\]
Coalgebraic semantics
De Simone rules

\[
\sigma(P) : -B \in \mathbb{P} \quad \gamma \text{ iso} \\
\frac{\quad P \xrightarrow{\gamma \circ \sigma} \gamma(B) \quad}{\quad G \xrightarrow{\sigma} B \quad (\tau, \sigma, \sigma', \tau') \text{ is a pushout} \\
\quad \frac{\quad \tau(G) \xrightarrow{\sigma'} \tau'(B) \quad}{\quad G \xrightarrow{\sigma} B \\
\quad G \cdot G' \xrightarrow{\sigma} B \cdot \sigma(G') \quad} \quad \frac{\quad G \xrightarrow{\sigma} B \\
\quad G' \cdot G \xrightarrow{\sigma} \sigma(G') \cdot B}\
\]
Induced coalgebra

(terms of \(\Pi\)-substitutive monoids)

(no axioms)

\[
\begin{array}{c}
\node{T_\Sigma} \downarrow p \\
\node{\mathcal{B}(T_\Sigma)}
\end{array}
\]

\[
P(t) = \left\{ (\sigma, t') : t \xrightarrow{\sigma} t' \text{ is a derivable sequent} \right\}
\]

\[
G \xrightarrow{\sigma} G'
\]

\[
\mathcal{P} \models G \Rightarrow_\sigma G'
\]
Homomorphism

\[ T_\Sigma \xrightarrow{\pi} G \]

(surjective)  (terms up-to axioms)

\[ \mathcal{B}(T_\Sigma) \]

(goal algebra)
Lifting

\[
\begin{array}{ccc}
T_\Sigma & \xrightarrow{\pi} & G \\
\downarrow^p & & \\
\mathcal{B}(T_\Sigma) & \xrightarrow{\mathcal{B}(\pi)} & \mathcal{B}(G)
\end{array}
\]
Structured coalgebra

\[ T_\Sigma \xrightarrow{\pi} \mathbb{G} \]

\[ \mathcal{B}(T_\Sigma) \xrightarrow{\mathcal{B}(\pi)} \mathcal{B}(\mathbb{G}) \]

(axioms bisimulate)
implies
(weak unique coalgebra)
Coalg(\mathcal{B}) has a final object
Examples
Example

\[
P(x, y, z) : - Q(x, y), R(y, z) \quad S(x, y, z) : - T(x, y, z)
\]
\[
Q(a, c) : - \quad T(a, c, z) : - V(z)
\]
\[
R(c, b) : - \quad T(x, c, b) : - U(x)
\]
\[
V(b) : - \quad U(a) : -
\]

\[P \sim S\]

isomorphic LTSs
Example

\[ P(f(x)) : - P(x) \quad Q(f(x)) : - R(x) \quad R(f(x)) : - Q(x) \]

\[ P \sim Q \sim R \]

perpetual processes
Some References


Conclusion

Algebraic structure of goals: substitutive monoids
Operational semantics: structured coalgebra
Bisimilarity is a congruence