07 Equational Logic and Algebraic Reasoning

Instructor: W. M. Farmer

Revised: 17 November 2004
What is Equational Logic?

- **Equational logic** is first-order logic restricted to languages with no predicate symbols except $\equiv$.

- An **equational theory** is a theory $T = (L, \Gamma)$ of equational logic such that each $A \in \Gamma$ is a universal closure of an equation of $L$, i.e., a closed formula of the form
  \[ \forall x_1, \ldots, x_n \cdot s = t. \]

- **Universal algebra** is the study of the models of equational theories.

- **Example.** The usual theory of groups is an equational theory.
Term Algebras

• A **ground term** is a variable-free term.

• The **term algebra** of a language $L = (C, F, P)$ of FOL with $C = \{c_1, \ldots, c_m\} \neq \emptyset$ and $F = \{f_1, \ldots, f_n\}$ is the algebra

  $$(D, c_1, \ldots, c_m, f_1, \ldots, f_n)$$

  where $D$ is the set of ground terms of $L$.

• A **term model** of a theory $T = (L, \Gamma)$ of FOL is a model of $T$ constructed from the term algebra of $L$. 
Initial Models

• An initial model of a theory $T = (L, \Gamma)$ of FOL is a model $M = (D, I)$ of $T$ such that:

  1. $M$ has no junk, i.e., for every $d \in D$, there is a ground term of $L$ whose value in $M$ is $d$.
  2. $M$ has no confusion, i.e., for all ground terms $s$ and $t$, $s = t$ is true in $M$ iff $s = t$ is valid in $T$.

• Theorem. Every equational theory has a unique initial model.
  – The initial model is a term model whose domain elements are equivalence classes of ground terms.

• The initial model semantics is often used in software engineering and computer science for equational theories instead of first-order semantics.
Example: Natural Numbers (1)

- Let $L = (\{0\}, \{S\}, \{=\})$ be a language of FOL where $S$ is unary.

- Consider the theory $T_1 = (L, \emptyset)$.
  - $T_1$ is an equational theory.

- The initial model of $T_1$ is $(\{0, S(0), S(S(0)), \ldots\}, 0, S)$.
  - Is the term algebra of $L$.
  - Represents the natural numbers.

- The other models of $T_1$ have junk or confusion.
  - Include both finite and infinite models.
  - Some contain “infinite” numbers.
Example: Natural Numbers (2)

- **Peano Arithmetic (1889).** \( T_2 = (L, \Gamma) \) where \( \Gamma \) contains the following three formulas:
  1. **0 has no predecessor.** \( \forall x . \neg (0 = S(x)) \)
  2. **S is injective.** \( \forall x, y . S(x) = S(y) \Rightarrow x = y \)
  3. **Induction principle.**
     \[ \forall P . (P(0) \land (\forall x . P(x) \Rightarrow P(S(x)))) \Rightarrow \forall x . P(x) \]

- **Theorem (Dedekind, 1888)** \( \{0, S(0), S(S(0)), \ldots\}, 0, S \) is the unique model of \( T_2 \) (up to isomorphism). That is, \( T_2 \) is **categorical**.

- The functions \( + \) and \( \ast \) can be defined in \( T_2 \).

- \( T_2 \) cannot be directly formalized in first-order logic.
Example: Natural Numbers (3)

- **Peano Arithmetic in FOL.** $T_3 = (L, \Gamma)$ is the theory of FOL where $\Gamma$ contains the following infinite set of formulas:
  1. **0 has no predecessor.** $\forall x . \neg(0 = S(x))$
  2. **S is injective.** $\forall x, y . S(x) = S(y) \Rightarrow x = y$
  3. **Induction schema.**
     
     $(A[x \mapsto 0] \land (\forall x . A \Rightarrow A[x \mapsto S(x)])) \Rightarrow \forall x . A$
     
     for each formula $A$ of $L$.

- $\{0, S(0), S(S(0)), \ldots\}, 0, S$ is the **standard model** of $T_3$.

- **Theorem.** $T_3$ has nonstandard models. This is, $T_3$ is **not categorical**.
  - The proof is by the compactness theorem.

- The functions $+$ and $\ast$ cannot be defined in $T_3$. 
Specifications and Descriptions

- A **specification** of a system $S$ can be formalized as a theory $T$ in some logic.
  - Each model of $T$ represents an **implementation** of $S$.
  - First-order logic and simple type theory are good for specifying systems.

- A **description** of a system $S$ can be formalized as a categorical theory $T$ in some logic.
  - The unique model of $T$ represents $S$ (up to isomorphism).
  - Equational logic with the initial model semantics and simple type theory are good for describing systems, but first-order logic is often not good.
Algebraic Reasoning

• Let $L$ be a language of FOL and $\mathcal{T}$ be the set of terms of $L$.

• Let $\mathcal{R}$ be a set of functions $r : \mathcal{T} \rightarrow \mathcal{T}$ called computation rules of $L$.
  
  $\quad r \in \mathcal{R}$ is sound if $t = r(t)$ for all $t \in \mathcal{T}$ with $r(t) \downarrow$.

• A computation in $\mathcal{R}$ is a finite sequence $C = \langle t_1, \ldots, t_n \rangle$ of terms of $L$ such that, for all $i$ with $1 \leq i < n$, there is some $r \in \mathcal{R}$ such that $t_{i+1} = r(t_i)$.

• Proposition. Suppose each $r \in \mathcal{R}$ is sound. If $\langle t_1, \ldots, t_n \rangle$ is a computation in $\mathcal{R}$, then $t_1 = t_n$. 
Substitutions

- Let $L = (C, F, P)$ be a language of FOL and $T$ be the set of terms of $L$.

- A substitution of $L$ is a total function $\sigma : V \rightarrow T$.

- The application of a substitution $\sigma$ to an expression $E$ of $L$, written $E\sigma$, is defined by recursion as follows:
  - If $x \in V$, $x\sigma = \sigma(x)$.
  - If $c \in C$, $c\sigma = c$.
  - If $f \in F$ is $n$-ary and $t_1, \ldots, t_n$ are terms of $L$, then $f(t_1, \ldots, t_n)\sigma = f(t_1\sigma, \ldots, t_n\sigma)$.
  - If $p \in P$ is $n$-ary and $t_1, \ldots, t_n$ are terms of $L$, then $p(t_1, \ldots, t_n)\sigma = p(t_1\sigma, \ldots, t_n\sigma)$.
  - If $A$ and $B$ are formulas of $L$, then $(\neg A)\sigma = \neg(A\sigma)$ and $(A \Rightarrow B)\sigma = (A\sigma \Rightarrow B\sigma)$.
  - If $x \in V$ and $A$ is a formula of $L$, then $(\forall x . A)\sigma = (\forall x . A\sigma')$ where $\sigma'(x) = x$ and $\sigma'(y) = \sigma(y)$ for all $y \neq x$. 
Matching

• Let $s$ and $t$ be terms and $A$ and $B$ be formulas of a language $L$ of FOL.

• $s$ matches $t$ if there is a substitution $\sigma$ of $L$ such that $s = t\sigma$. Similarly, $A$ matches $B$ if there is a substitution $\sigma$ of $L$ such that $A = B\sigma$.

• If $s$ matches $t$, then $s$ is an instance of the pattern $t$.

• Matching is used in many places in CS and SE, e.g., in term rewriting.
Unification

• Let $s$ and $t$ be terms and $A$ and $B$ be formulas of a language $L$ of FOL.

• $s$ and $t$ unify if there is a substitution $\sigma$ of $L$ such that $s\sigma = t\sigma$. Similarly, $A$ and $B$ unify if there is a substitution $\sigma$ of $L$ such that $A\sigma = B\sigma$.

• Unification is solving equations by syntax alone.

• The unifying substitution is called a unifier.

• Unification is used in many places in CS and SE, e.g., in resolution theorem proving and logic programming.
Most General Unifiers

• A **most general unifier** of \( s \) and \( t \) is a unifier \( \sigma \) of \( s \) and \( t \) such that, if \( \sigma' \) is a unifier of \( s \) and \( t \), then there is substitution \( \tau \) such that, for all \( x \in \mathcal{V} \),

\[
x\sigma' = (x\sigma)\tau.
\]

• **Theorem.** For every pair of terms \( s \) and \( t \) of \( L \), if \( s \) and \( t \) are unifiable, there is a most general unifier of \( s \) and \( t \) that is unique up to a renaming of variables.

• The first algorithm to compute the most general unifier of two first-order terms was given by Herbrand in 1930.
Equational Reasoning

• In an equational theory $T = (L, \Gamma)$, the fundamental rules of inference are substitution and replacement:
  
  – If $T \models s = t$, then $T \models s\sigma = t\sigma$ for every substitution $\sigma$ of $L$.
  – If $T \models s = t$, then $T \models u = u'$ where $u'$ is obtained by replacing one occurrence of $s$ in $u$ by $t$.

• Problem: How do we choose the substitutions?
Term Rewriting Systems

• Let $L$ be a language of FOL and $T$ be the set of terms of $L$.

• A **rewrite rule** of $L$ is a directed equation of $L$, written $s \rightarrow t$, such that all the variables of $t$ are contained in $s$.

• A **term rewriting system** of $L$ is a set $\mathcal{R}$ of rewrite rules of $L$.

• The **reduction relation** $\rightarrow_{\mathcal{R}} \subseteq T \times T$ is the smallest relation containing $\mathcal{R}$ and closed under **substitution** and replacement:
  
  - If $s \rightarrow_{\mathcal{R}} t$, then $s\sigma \rightarrow_{\mathcal{R}} t\sigma$ for every substitution $\sigma$ of $L$.
  - If $s \rightarrow_{\mathcal{R}} t$, then $u \rightarrow_{\mathcal{R}} u'$ where $u'$ is obtained by replacing one occurrence of $s$ in $u$ by $t$. 

Other Relations

$\rightarrow^*: \text{the reflexive-transitive closure of } \rightarrow_R$.

$\leftrightarrow_R: \text{the symmetric closure of } \rightarrow_R$.

$\leftrightarrow^*_R: \text{the reflexive-symmetric-transitive closure of } \rightarrow_R$. 
Soundness and Completeness

- Let $T = (L, \Gamma)$ be an equational theory and $\mathcal{R}$ be a term rewriting system for $L$.

- $\mathcal{R}$ is **sound** with respect to $T$ if

  $$s \rightarrow_{\mathcal{R}} t \ \text{implies} \ T \models s = t.$$ 

- $\mathcal{R}$ is **complete** with respect to $T$ if

  $$T \models s = t \ \text{implies} \ s \leftrightarrow_{\mathcal{R}}^* t.$$ 

- **Proposition.** For each $s = t \in \Gamma$, assume all the variables of $t$ are contained in $s$. Then $\mathcal{R} = \{s \rightarrow t \mid s = t \in \Gamma\}$ is a term rewrite system of $L$ which is sound and complete with respect to $T$. 


Norm Forms

• Let $\mathcal{R}$ be a term rewriting system for $L$.

• A term $s$ of $L$ is in **normal form** relative to $\mathcal{R}$ if there is no term $t$ such that $s \rightarrow^\ast_{\mathcal{R}} t$.
  
  – That is, no subterm of $s$ matches the left side of a rewrite rule in $\mathcal{R}$.

• $t$ is a **normal form** of $s$ relative to $\mathcal{R}$ if $s \rightarrow^\ast_{\mathcal{R}} t$ and $t$ is in normal form relative to $\mathcal{R}$.
Properties of Term Rewriting Systems

- Let $\mathcal{R}$ be a term rewriting system for $L$.

- $\mathcal{R}$ is **Church-Rosser** if, for all terms $s, t$ of $L$, $s \leftrightarrow^*_\mathcal{R} t$ iff there exists some $u$ such that $s \rightarrow^*_\mathcal{R} u$ and $t \rightarrow^*_\mathcal{R} u$.

- $\mathcal{R}$ is **confluent** if, for all terms $s, t, u$ of $L$, $u \rightarrow^*_\mathcal{R} s$ and $u \rightarrow^*_\mathcal{R} t$ implies there is some term $v$ such that $s \rightarrow^*_\mathcal{R} v$ and $t \rightarrow^*_\mathcal{R} v$.

- $\mathcal{R}$ is **noetherian** or **finitely terminating** if there is no infinite chain of reductions

  $$s_1 \rightarrow^\mathcal{R} s_2 \rightarrow^\mathcal{R} s_3 \rightarrow^\mathcal{R} \cdots.$$
Theorems of Term Rewriting Systems

• Let $\mathcal{R}$ be a term rewriting system for $L$.

• **Theorem.** $\mathcal{R}$ is Church-Rosser iff $\mathcal{R}$ is confluent.

• **Proposition.** If $\mathcal{R}$ is confluent, then the normal form of a term of $L$ is unique when it exists.

• **Proposition.** If $\mathcal{R}$ is finitely terminating, then every term of $L$ has a normal form.

• **Theorem.** Let $T = (L, \Gamma)$ be an equational theory and $\mathcal{R}$ be sound and complete with respect to $T$, finite, confluent, and finitely terminating. Then:
  1. Every term $s$ of $L$ has a unique normal form $t$ relative to $\mathcal{R}$ such that $T \models s = t$.
  2. It is decidable whether $T \models s = t$. 


Knuth-Bendix Completion Algorithm

- Let $T = (L, \Gamma)$ be an equational theory such that $\Gamma$ is finite.

- Given $\Gamma$ and a reduction order as input, the Knuth-Bendix completion algorithm does one of the following:
  1. Returns a term rewriting system $\mathcal{R}$ for $L$ that is sound and complete with respect to $T$, finite, confluent, and finitely terminating.
  2. Terminates with failure.
  3. Loops without terminating.

- The algorithm is composed of two steps:
  1. Creation of an initial set of rules by orienting the members of $\Gamma$ according to the reduction order.
Restricted Systems of First-Order Logic

• A **conditional equation** is a formula of the form

\[ A \Rightarrow s = t. \]

- Are used as conditional rewrite rules.

• A **Horn clause** is a formula of the form

\[ A_1 \land \cdots \land A_n \Rightarrow B \]

where \( A_1, \ldots, A_n, B \) are positive literals and \( n \geq 0 \).

- A Horn clause of the form \( B \) is called a **goal**.

• Computation in restricted systems:

<table>
<thead>
<tr>
<th>Kind of Formulas</th>
<th>Kind of Computation</th>
</tr>
</thead>
<tbody>
<tr>
<td>equations</td>
<td>term rewriting</td>
</tr>
<tr>
<td>conditional equations</td>
<td>constraint programming</td>
</tr>
<tr>
<td>Horn clauses</td>
<td>logic programming</td>
</tr>
</tbody>
</table>