Constructor-based Logics

GAINA, Daniel

Japan Advanced Institute of Science and Technology

January 22, 2010

Introduction

- Algebras consist of sorted-sets + functions
- Programs as algebras

An overview

- Signatures : sort and operation names
- Equations : describe the behavior of operations
- Signatures + Equations = (Basic) specifications
- Algebras of specifications interpret
  1. each sort name as a set,
  2. each operation name as a function, and
  3. satisfy the equations.
- Specifications describe the behavior of algebras/programs

Specification of natural numbers

```
mod* PNAT {
[Nat]
op 0 :  -> Nat
op s_ :  Nat -> Nat
op _+_ :  Nat Nat -> Nat
vars X Y : Nat
eq [ladd1] :  0 + X = X .
eq [ladd2] :  s X + Y = s (X + Y) . }
```

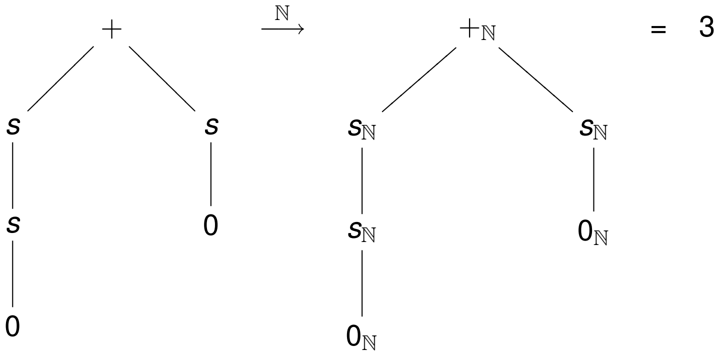
- $S = \{Nat\}$, $F = \{0, s\_, \_+\_\}$
- `ladd1` and `ladd2` are (S, F)-equations
- $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Z}_n$ are (S, F)-models

Remark

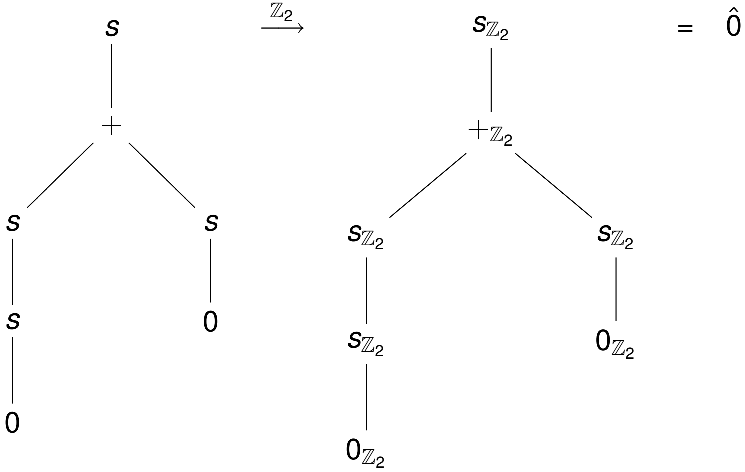
Logical notation: $\forall X.0 + X = X$
CafeOBJ notation: `eq 0 + X = X .`
(Note that `X` is previously declared as a variable)

Terms and models I

Models interpret a ground term uniquely.

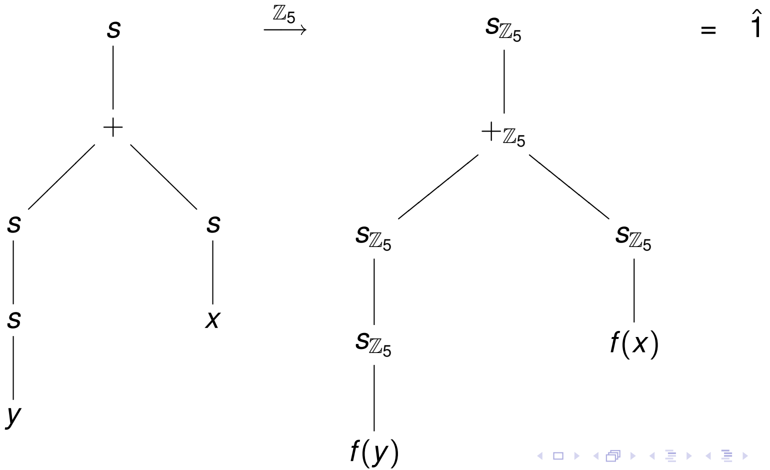


Terms and models II



Terms and models III

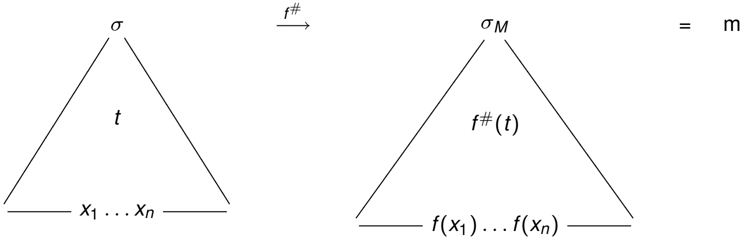
$f : \{x, y\} \to \mathbb{Z}_5$, $f(x) = \hat{3}$ and $f(x) = \hat{4}$



Terms and models IV

1. algebra (S, F)-algebra M, and
2. an assignment $f : \{x_1, \ldots, x_n\} \to M$.

Terms t with var. $\{x_1, \ldots, x_n\}$ are uniquely interpreted (via f) in M



$T_F(\{x_1, \ldots, x_n\})$ is the (S, F)-algebra of terms with var. $\{x_1, \ldots, x_n\}$.

Terms and models V

Proposition

$f : \{x_1, \ldots, x_n\} \to M$ can be uniquely extended to terms with variables from $\{x_1, \ldots, x_n\}$, $f^\# : T_F(\{x_1, \ldots, x_n\}) \to M$

$$\begin{tikzcd}
T_F(\{x_1, \ldots, x_n\}) \arrow[r, "\exists! f^\#"] & M \\
\{x_1, \ldots, x_n\} \arrow[u, hook] \arrow[ur, "f"'] &
\end{tikzcd}$$

Satisfaction I

M is (S, F)-algebra

- $(\forall X)t = t'$ is (S, F)-equation
  $M \models_{(S,F)} (\forall X)t = t' \Leftrightarrow f^{\#}(t) = f^{\#}(t')$,
  for all $f : X \to M$.

- $M \models_{(S,F)} (\forall X)t = t'$ if $b \Leftrightarrow$
  $f^{\#}(b) = true_M \Rightarrow f^{\#}(t) = f^{\#}(t')$,
  for all $f : X \to M$.

Satisfaction II

Remark

If $X = \{x, y, z\}$ then $(\forall X)e = (\forall x)(\forall y)(\forall z)e$.

We may drop the subscript (S, F) from $\models_{(S,F)}$ when it is understood from the context.

Satisfaction III

- $S = \{Nat\}$ $F = \{0, s, +\}$
- $\mathbb{N}$ is (S, F)-model
- $(\forall x)(\forall y)x + sy = s(x + y)$ is (S, F)-equation
- $f : \{x, y\} \to \mathbb{N}$, $f(x) = 3$ and $f(y) = 7$
- $f^{\#}(x + sy) = f(x) +_{\mathbb{N}} s_{\mathbb{N}} f(y) = 3 +_{\mathbb{N}} s_{\mathbb{N}} 7 = 3 +_{\mathbb{N}} 8 = 11$
- $f^{\#}(s(x + y)) = s_{\mathbb{N}}(f(x) +_{\mathbb{N}} f(y)) = s_{\mathbb{N}}(3 +_{\mathbb{N}} 7) = s_{\mathbb{N}} 10 = 11$

Remark

$\mathbb{N} \models (\forall x)(\forall y)x + sy = s(x + y)$ iff $f^{\#}(x + sy) = f^{\#}(s(x + y))$, for all $f : X \to \mathbb{N}$

Specifications and models

Definition

Let $Sp = ((S, F), E)$ be a specification.

1. We say that M is a model of the specification Sp if $M \models E$ (i.e. $M \models e$ for all $e \in E$).
2. $E \models e$ iff $M \models e$ for all models M of the specification Sp. In this case we may write $Sp \models e$.

Exercise: Show that $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Z}_2$ are models of the specification `PNAT`.

What is verification about?

- Some programs (models of the specifications) satisfies some properties (written as equations).
- The only effective way to prove formally the truth is by syntactic means (using proof rules).

Proof rules

1. *Reflexivity*: $\dfrac{}{\emptyset \vdash_\Sigma t = t}$
2. *Symmetry*: $\dfrac{}{t = t' \vdash_\Sigma t' = t}$
3. *Transitivity*: $\dfrac{}{\{t = t',\ t' = t''\} \vdash_\Sigma t = t''}$
4. *Congruence*: $\dfrac{}{t = t' \vdash_\Sigma t_0(z \leftarrow t) = t_0(z \leftarrow t')}$
5. *Substitutivity*: $\dfrac{}{(\forall x)e \vdash_\Sigma (\forall Y)e(x \leftarrow t)}$, where $t \in T_\Sigma(Y)$
6. *Implications*: $\dfrac{E \vdash_\Sigma t = t' \text{ if } b}{E \cup \{b = \mathit{true}\} \vdash t = t'}$ and $\dfrac{E \cup \{b = \mathit{true}\} \vdash_\Sigma t = t'}{E \vdash_\Sigma t = t' \text{ if } b}$
7. *Generalization*: $\dfrac{E \vdash_\Sigma (\forall x)e}{E \vdash_{\Sigma(x)} e}$ and $\dfrac{E \vdash_{\Sigma(x)} e}{E \vdash_\Sigma (\forall x)e}$

Proof properties

1. *Monotonicity*: $\dfrac{}{E \vdash_{\Sigma} E'}$ if $E' \subseteq E$
2. *Transitivity*: $\dfrac{E_1 \vdash_{\Sigma} E_2, E_2 \vdash_{\Sigma} E_3}{E_1 \vdash_{\Sigma} E_3}$
3. *Unions*: $\dfrac{E \vdash_{\Sigma} E_1, E \vdash_{\Sigma} E_2}{E \vdash_{\Sigma} E_1 \cup E_2}$

Equational proof I

$E \subseteq \mathbb{S}en(\Sigma)$ is a set of sentences and $e \in \mathbb{S}en(\Sigma)$ is a sentence, where $\Sigma = (S, F)$. A proof of e from E, written as $E \vdash_\Sigma e$, is a (finite) sequence of goals $E_1 \vdash_\Sigma e_1, \ldots E_n \vdash_\Sigma e_n$ such that

1. $E_n = E$ and $e_n = e$
2. $E_{i+1} \vdash_\Sigma e_{i+1}$ is obtained by applying a proof rule/property to the subset of $\{E_1 \vdash_\Sigma e_1, \ldots E_i \vdash_\Sigma e_i\}$

We may drop the subscript from $E \vdash_\Sigma e$ and write $E \vdash e$ when there is no danger of confusion.

Proposition (Soundness)

$E \vdash e_n$ implies $E \models e_n$

Equational proof II

```
mod* GROUP {
[Group]
op 0 :  -> Group
op _+_ :  Group Group -> Group
op -_ :  Group -> Group
vars X Y Z : Group
eq [lid] :  0 + X = X .
eq [linv] :  (- X) + X = 0 .
eq [assoc] :  X + (Y + Z) = (X + Y) + Z . }
```

Equational proof III

- $\Sigma = (S, F)$
  $S = \{\texttt{Group}\}$
  $F = \{\texttt{0}, \texttt{+}, \texttt{-}\}$
  $E = \{\texttt{lid, linv, assoc}\}$

- Assume
  `eq [rinv] : X + (-X) = 0 .`
  and prove
  `eq [rid] : X + 0 = X .`

- By *Generalization*:
  $E \vdash_{\Sigma} (\forall \mathrm{X})\texttt{X+0=X}$ iff $E \vdash_{\Sigma(a)} \texttt{a+0=a}$,
  where `a` is a any constant of sort `Group`.

Equational proof IV

1. `0+a=a` by `lid` for `X` substituted by `a`
2. `a+(-a)=0` by `rinv` for `X` substituted by `a`
3. `(a+(-a))+a=0+a` by *Congruence* with t_0=`z+a`
4. `(a+(-a))+a=a` by *Transitivity* applied to 3 and 1
5. `a+(-a+a)=(a+(-a))+a` by `assoc` for `X=a`, `Y=-a` and `Z=a`
6. `a+(-a+a)=a` by *Transitivity* applied to 5 and 4
7. `-a+a=0` by `linv` for `X=a`
8. `0=-a+a` by *Symmetry*
9. `a+0=a+(-a+a)` by *Congruence* with t_0=`a+z`
10. `a+0=a` by *Transitivity* applied to 9 and 6

Subterm replacement

- Specification (Σ, E) with $\Sigma = (S, F)$
  Substitution $\theta : \{x\} \rightarrow T_F(Y)$
  Conditional equation $(\forall x)t = t'$ if b in E

- *Subterm replacement:* $\dfrac{E \vdash \theta(b) = true}{E \vdash t_0(z \leftarrow \theta(t_1)) = t_0(z \leftarrow \theta(t_2))}$

CafeOBJ proofs

Assume `GROUP` satisfies `eq [rinv] : X + (-X) = 0`
and prove `eq [rid] : X + 0 = X`

```
open GROUP + EQL
vars X Y Z : Group .
eq [rinv] : X + (- X) = 0 .
op a : -> Group .
start a + 0 = a .
apply -.linv with X = a at (1 2) .
**> result a + (- a + a) = a : Bool
apply assoc at (1) .
**> result (a + (- a)) + a = a : Bool
apply red at term .
**> result true : Bool
close
```

A model of NAT

$\mathbb{N}'$ model interpreting

1. sort `Nat`:
   $\mathbb{N} \cup \{0'\}$
2. function `s_`:
   $\mathtt{s}_{\mathbb{N}'}\ \mathtt{0'} = \mathtt{1}$
3. function `_ + _`:
   $\mathtt{0'} +_{\mathbb{N}'} \mathtt{0} = \mathtt{0'}$
   $\mathtt{0'} +_{\mathbb{N}'} \mathtt{X} = \mathtt{X}$ for all $\mathtt{X} \in \mathbb{N} - \{0\}$
   $\mathtt{X} +_{\mathbb{N}'} \mathtt{0'} = \mathtt{X}$ for all $\mathtt{X} \in \mathbb{N}$

Remark

$\mathtt{0} +_{\mathbb{N}'} \mathtt{0'} \neq \mathtt{0'} +_{\mathbb{N}'} \mathtt{0}$

Constructor-based signatures

(S, F, F^c) signature: (S, F) - algebraic signature
$F^c \subseteq F$ - constructors

1. constrained sorts = sorts of constructors
2. a sort which is not constrained is loose

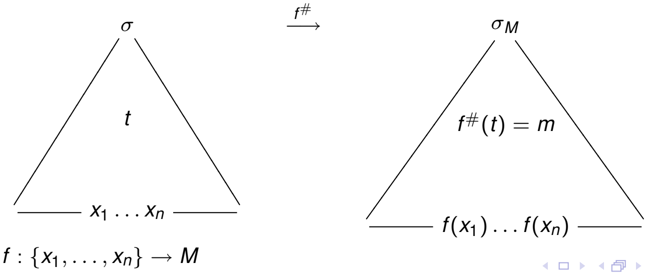
```
mod* PNAT {
[Nat]
op 0 :  -> Nat {constr}
op s_ :  Nat -> Nat {constr}
op _+_ :  Nat Nat -> Nat
vars X Y : Nat
eq 0 + X = X .
eq s X + Y = s (X + Y) . }
```

Remark

constrained sorts $\{Nat\}$
loose sorts $\emptyset$

Constructor-based models I

Constructor-based algebras M consist of interpretation of constructor terms formed with constructors and elements of sort loose, i.e. for every element m of constrained sort there exists a constructor term $t \in T_{F^c}(\{x_1, \ldots, x_n\})$, where variables x_i have loose sorts, and an assignment $f : \{x_1, \ldots, x_n\} \to M$ such that $f^{\#}(t) = m$.



Constructor-based models II

for $\mathtt{NAT}$: no loose sorts

- $\mathbb{N}$ constructor-based algebra
  for all $n \in \mathbb{N}$ there is $\mathtt{s \ldots s\ \ 0}$ such that $\mathtt{s}_{\mathbb{N}} \ldots \mathtt{s}_{\mathbb{N}} \mathtt{0}_{\mathbb{N}} \mathtt{=n}$
- $\mathbb{Z}_2$ constructor-based algebra
  $\mathtt{0}_{\mathbb{Z}_2} = \mathtt{0}$ and $\mathtt{s}_{\mathbb{Z}_2} \mathtt{0}_{\mathbb{Z}_2} \mathtt{=1}$
- $\mathbb{Z}$ is not constructor-based algebra
  there exists no term $\mathtt{s \ldots s\ \ 0}$ such that $\mathtt{s}_{\mathbb{N}} \ldots \mathtt{s}_{\mathbb{N}} \mathtt{0}_{\mathbb{N}} \mathtt{=-1}$
- $\mathbb{N}'$ is not constructor-based algebra
  there is no term $\mathtt{s \ldots s\ \ 0}$ s. t. $\mathtt{s}_{\mathbb{N}} \ldots \mathtt{s}_{\mathbb{N}} \mathtt{0}_{\mathbb{N}} \mathtt{=0'}$

Induction

- (Σ, E) specification

  $(\forall x)e$ conditional equation, x is constrained var.

  $$\textit{Induction:}\ \frac{\{E \vdash^c (\forall Y)e(x \leftarrow t) \mid t \text{ constructor term}\}}{E \vdash^c (\forall x)e}$$

- (Σ, E)=$\mathtt{NAT}$ and $e = (\forall y)x + sy = s(x + y)$

  By *Induction* we need to prove

  1. $(\forall y)0 + sy = s(0 + y)$
  2. $(\forall y)s0 + sy = s(s0 + y)$
  3. $(\forall y)ss0 + sy = s(ss0 + y)$

$\vdots$

Induction scheme

IB $E \vdash^{c}_{\Sigma} (\forall y) 0 + sy = s(0 + y)$

IS $E \cup (\forall y) a + sy = s(a + y) \vdash^{c}_{\Sigma(a)} (\forall y) sa + sy = s(sa + y)$

CafeOBJ code:

IB
```
open PNAT
red 0 + s Y = s(0 + Y) .
close
```

IS
```
open PNAT
op a :  -> Nat .
eq [IH] : a + s Y = s(a + Y) .
red s a + s Y = s(s a + Y) .
close
```

Equality _=_

```
mod* SPEC {
[Elt]
ops a b :  -> Elt
op _=_ :  Elt Elt -> Bool
vars X Y : Elt
eq [equal] :  (X = X) = true .
ceq [cequal] :  X = Y if (X = Y) .
}
```

Lemma (Equality)

We have that

1. $\{$equal, cequal, a=b$\} \vdash^{c}$ (a=b)=true
2. $\{$equal, cequal, (a=b)=true$\} \vdash^{c}$ a=b

Inconsistency I

Definition

(Σ, E) is inconsistent if $E \vdash^{c} \mathit{true} = \mathit{false}$

Remark

(Σ, E) admits initial model even it is inconsistent.

Inconsistency II

Proposition (Inconsistency)

Assume

```
mod* SP{
inc(SPEC)
eq a = b .
eq (a = b) = false .  }
```

Then SP *is inconsistent and* SP$\vdash^{c} \forall x.\forall y.x = y$.

Inconsistency III

```
mod* QUEUE(D :: SPEC){
[Queue]
op nil : -> Queue {constr}
op _@_ : Elt Queue -> Queue {constr}
op _in_ : Elt Queue -> Bool
op empty? : Queue -> Bool
var Q : Queue
vars X Y : Elt
eq X in nil = false .
eq X in (Y @ Q) = (X = Y) or (X in Q) .
eq empty?(nil) = true .
eq empty?(X @ Q) = false . }
```

Inconsistency IV

The following spec. is inconsistent

```
mod* QUEUE-I{
inc(QUEUE)
op q :  -> Queue
eq a @ q = nil .  }
```

We would have

```
false = empty?(a @ q) = empty?(nil) = true
```

Case analysis I

- (Σ, E), specification with $\Sigma = (S, F, F^c)$.
- $\sigma \in (F - F^c)$ operation of constrained sort s
- $t_1, \ldots, t_n$ constructor terms
- $\sigma(t_1, \ldots, t_n)$ is "not defined", there is no constructor term t such that $E \models^c \sigma(t_1, \ldots, t_n) = t$.

Case analysis: $$\frac{\{E \cup \{\sigma(t_1, \ldots, t_n) = t\} \vdash^c e \mid t \text{ constructor term}\}}{E \vdash^c e}$$

```
mod* SPEC-CA{
[Elt]
op a :  -> Elt {constr}
op b :  -> Elt }
```

We have that `SPEC-CA` $\vdash^c$ `a = b`.

Case analysis II

```
mod* QUEUE={
inc(QUEUE)
eq a @ nil = b @ nil .  }
```

Using *Case analysis*

```
eq (a = b) = true .
eq (a = b) = false .
```

one may easily prove that `QUEUE=` $\vdash^{C}$ `a = b`

Soundness

Proposition (Soundness)

$E \vdash^c e$ implies $E \models^c e$.

Remark

The proof rules of constructor-based equational logic are not sound for equational logic, i.e. $E \vdash^c e$ does not imply $E \models e$.

Consider $\mathtt{PNAT}$ and the model $\mathbb{N}'$ of $\mathtt{PNAT}$.
We have $\mathtt{PNAT} \vdash^c \forall x.\forall y.x + y = y + x$ but
$\mathtt{PNAT} \not\models \forall x.\forall y.x + y = y + x$ because $\mathbb{N}'$ is a model of $\mathtt{PNAT}$ but
$\mathbb{N}' \not\models \forall x.\forall y.x + y = y + x$.

Exercises

Using CafeOBJ prove

1. `Group` $\models$ `rinv`
2. Lemma Equality
3. `QUEUE-I` $\models^{c} \forall x.\forall y.x = y$
4. `SPEC-CA` $\models^{c}$ `a = b`
5. `QUEUE=` $\models^{c}$ `a = b`