Models, Satisfactions, and Proof Rules for CafeOBJ Specifications

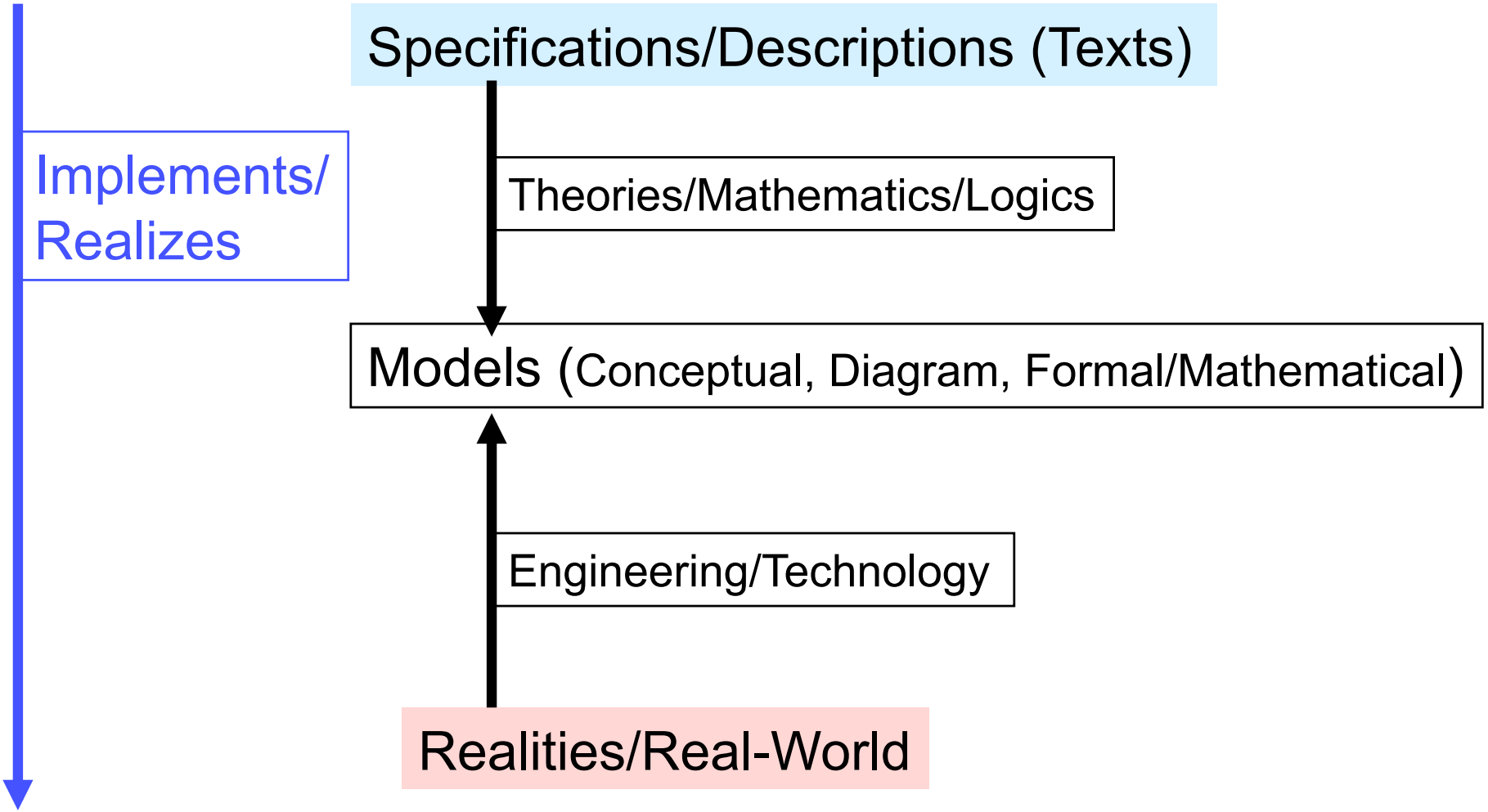
Lecture Note 07a

Formal Methods (i613-0912)

Topics

- Specification/Descriptions, Models, and Realities
- Order Sorted Term Algebra
- Satisfaction of a Property by a Specification
  - SPEC |= prop
- Proof rules for SPEC |= prop

Specifications, Models, Realities



Specification

An **constructor-based equational specification SPEC** in CafeOBJ (a legitimate text in the CafeOBJ language with only equations as axioms) is defined as a pair **(Sig,E)** of order-sorted constructor-based signature **Sig** and a set **E** of conditional equations over **Sig**. A signature **Sig** is defined as a triple **(S,F,F^c)** of an orded-sorted set **S** of sorts, a set **F** of functions/operations over **S**, and a set **F^c** of constructors. **F^c** is a subset of **F**, i.e. **Fc** $\subseteq$ **F** .

SPEC = ((S,F,F^c),E)

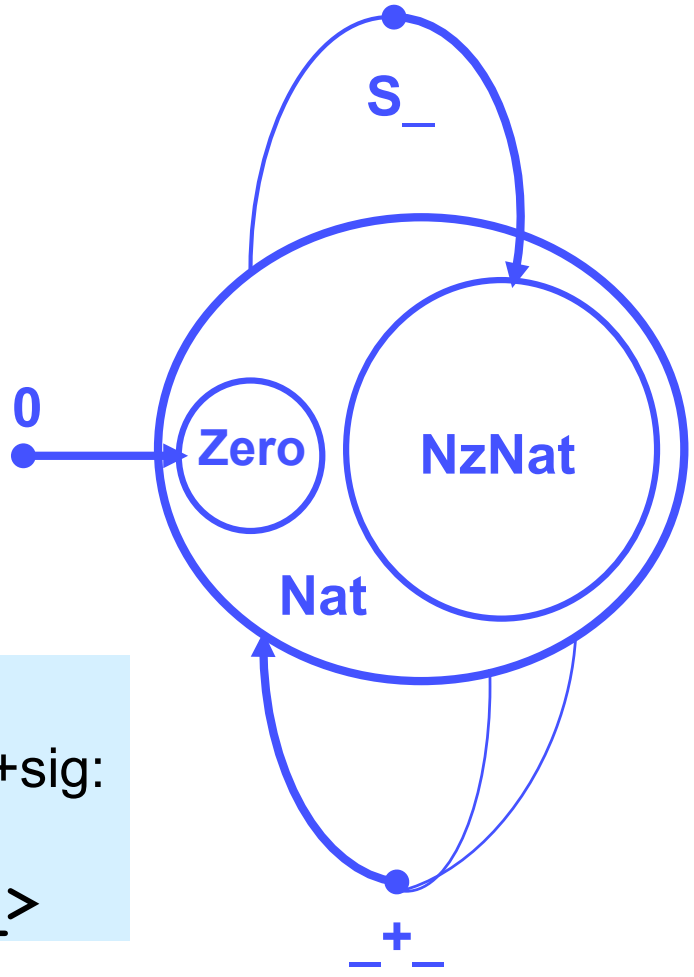
Model (Algebra)

A formal/mathematical **model** of a specification **SPEC = ((S,F,F^c),E)** is an order-sorted constructor-based **algebra A** (see p.23 of LectureNote05) which has the signature **(S,F,F^c)** and satisfies (see p.10 of LectureNote05) all equations in **E**.

An algebra which has a signature (S,F) is called an **(S,F)-algebra.** An **(S,F)-algebra A** interprets a sort symbol s in **S** as a (non empty) set $\mathbf{A_S}$ and an operation (function) symbol **f :s1 s2 …sn->s(n+1)** in **F** as a function $\mathbf{A_f : A_{s1},A_{s2},..,A_{sn}->_{As(n+1)}}$. The interpretation respects the order-sort constrains.

An example of Signature and its Algebra

```
-- signature NAT+sig
-- sort
[ Zero NzNat < Nat ]
-- operators
op 0 :   -> Nat {constr}
op s_ :  Nat -> NzNat {constr}
op _+_ : Nat Nat -> Nat
```



A MAT+sig-algebra
Order-Sorted Algebra with Signature NAT+sig:

<Nat, NzNat, Zero; 0, s_, _+_>

Order Sorted Term Algebra $T_{NAT+sig}$ of Signature NAT+sig

An *Order-Sorted Algebra* is a mathematical object composed of order-sorted carrier sets and operations over them.

Order-Sorted Carrier sets can be thought of Order-Sorted Sets of Terms:

```
Zero = { 0 }
NzNat = { s n | n ∈ Nat }
Nat = Zero ∪ NzNat ∪
      { n1 + n2 | n1 ∈ Nat ∧ n2 ∈ Nat }
```

Operations over the order-sorted sets of terms:

```
 0 = 0
(for n∈Nat)(s n = s n)
(for n1,n2∈Nat)(n1 + n2 = n1 + n2)
```

Valuation, evaluation, equation

A **valuation** (or an assignment) is a sort preserving map from the (order-sorted) set of variables of a specification to an order-sorted algebra (a model), and assigns values to all variables.

Given a model **A** and a valuation **v**, a **term t** of sort **s,** which may contain variables, is evaluated to a **value** $\mathbf{A_v(t)}$ in a set $\mathbf{A_s}$

Given terms $\mathbf{t_1}$ and $\mathbf{t_2}$ of a sort **s** and term **c** of sort **Bool**, a **conditional equation** is a sentence of the form:

$$\mathbf{t_1 = t_2 \text{ if } c}$$

An ordinary equation $\mathbf{t_1 = t_2}$ is an abbreviation of

$$\mathbf{t_1 = t_2 \text{ if true}}$$

Satisfiability of equation

Given a model (an ordered-sorted algebra) **A**,

A satisfies a conditional equation $t_1 = t_2$ if c

iff

$A_v(c)$ implies **$A_v(t_1) = A_v(t_2)$**

for any valuation **v** .

For an ordinary unconditional equation **$t_1 = t_2$** , **$A_v(c)$** is understood to be **true**, for **$t_1 = t_2$** is synonym of **$t_1 = t_2$ if true** .

The satisfaction of an equation by a model **A** is denoted by

A |e= ($t_1 = t_2$ if c) or **A |e= ($t_1 = t_2$)**

SPEC-algebra

For a CafeOBJ specification **SPEC** = (**(S,F,F^c), E)**, a **SPEC-algebra** is a (S,F)-algebra which satisfy

(1) satisfies all equations in **E**

and

(2) is a constructor-based algebra

Satisfiability of boolean term

A SPEC-algebra **A satisfies a term t of sort Bool** iff $\mathbf{A_v(t)}$ = **true** for any valuation **v** (or iff **A** satisfies an equation **t = true**) .

The satisfaction of a predicate by a model **A** is denoted by:

$$\mathbf{A \models p}$$

Only the satisfaction relation:

$$\mathbf{A \models p}$$

is simulated by CafeOBJ System. The satisfaction relation

$$\mathbf{A \models_e (t_1 = t_2)}$$

can not be simulated by CafeOBJ system, because equation is a meta-entity and not in the object level of CafeOBJ.

Equality predicate _=_

There is a special operation symbol `_=_` with an operation declaration `op (_=_) : s s -> Bool` for any sort symbol **s** of any signature **S** of any specification **SPEC = ((S,F,Fc),E)**. For any model **A**, `_=_` is **postulated** to be interpreted as the equality (or identity) relation on the set $\mathbf{A}_s$.

This is formulated as

```
eq X = X .
ceq (X = Y) if X = Y .
```

For any specification **SPEC = ((S,F,Fc),E)**, any **SPEC-algebra A** is postulated to satisfy:

$$\mathbf{A} \models (t_1 = t_2) \quad \text{iff} \quad \mathbf{A} \models_e (t_1 = t_2)$$

for any pair of terms t_1 and t_2 .

Satisfiability of property by specification: SPEC |= prop

A specification **SPEC** = **((S,F,F^c),E)** is defined to satisfy a property **p** (a term of sort **Bool)** iff **A |= p** holes for any **SPEC-algebra A.**

The satisfaction of a predicate **prop** by a specification **SPEC** = **((S,F,F^c),E)** is denoted by:

SPEC |= p or E |= p

A most important purpose of developing a specification **SPEC = ((S,F,F^c),E)** in CafeOBJ is to check whether

SPEC |= prop

holds for a predicate **prop** which describe some important property of the system which **SPEC** specifies.

Proof rules for SPEC |e= (t1 = t2) (p.14 of LN05)

Proof rules

(Σ, E) specification, where $\Sigma = (S, F)$

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:* $\dfrac{E \vdash_\Sigma t = t' \text{ if } b}{E \cup \{b = true\} \vdash t = t'}$ and $\dfrac{E \cup \{b = true\} \vdash_\Sigma t = t'}{E \vdash_\Sigma t = t' \text{ if } b}$
7. *Generalization:* $\dfrac{E \vdash_\Sigma (\forall x)e}{E \vdash_{\Sigma(x)} e}$ and $\dfrac{E \vdash_{\Sigma(x)} e}{E \vdash_\Sigma (\forall x)e}$

Some proof rules for SPEC |= prop (1)

(Induction) {E |- (∀Y)e(x←t) | t constructorTerm}
implies {E⊢(∀x)(∀Y)e}

```
{                    (INV1 |- mx(init)),
 (INV1∪{mx(s)=true} |- ∀k∈Pid.mx(want(s,k))),
 (INV1∪{mx(s)=true} |- ∀k∈Pid.mx(try(s,k))),
 (INV1∪{mx(s)=true} |- ∀k∈Pid.mx(exit(s,k))) }
                       implies
                    (INV1 |- ∀s∈Sys.mx(s))
```

Some proof rules for SPEC |= prop (2)

(CaseAnalysis) { E ∪ {σ(t1,...,tn)=t |- e | t constructorTerm}
implies E |- e

(CS) { (E |- (p_1 or p_2)), (E U {p_1=true} |- p) ,
(E U {p_2=true} |- p) }
implies E |- p

Some proof rules for SPEC |= prop (3)

(EQ) E U { $t_1 = t_2$ } |- p iff E U { (t1 = t2) = true } |- p

Let t1' and t2' be the terms obtained from terms t1 and t2 by replacing variables in t1 and t2 with corresponding ground terms respectively then:

(INST) (E U {t1=t2})|- p iff
(E U {t1=t2} U {t1'=t2'})|- p

(TRANS) E |- (($t_1 = t_2$) implies p) iff
E U { $t_1 = t_2$ } |- p

Some proof rules for SPEC |= prop (4)

(ConditionalRewriting)
{ E ∪ {t1=t2 if b} |- θ(b), E ∪ {t1=t2 if b} |- t0(z←θ(t1) }
Implies E ∪ {t1=t2 if b} |- t0(z←θ(t2)

CafeOBJ system applies Conditional Rewriting as much as possible