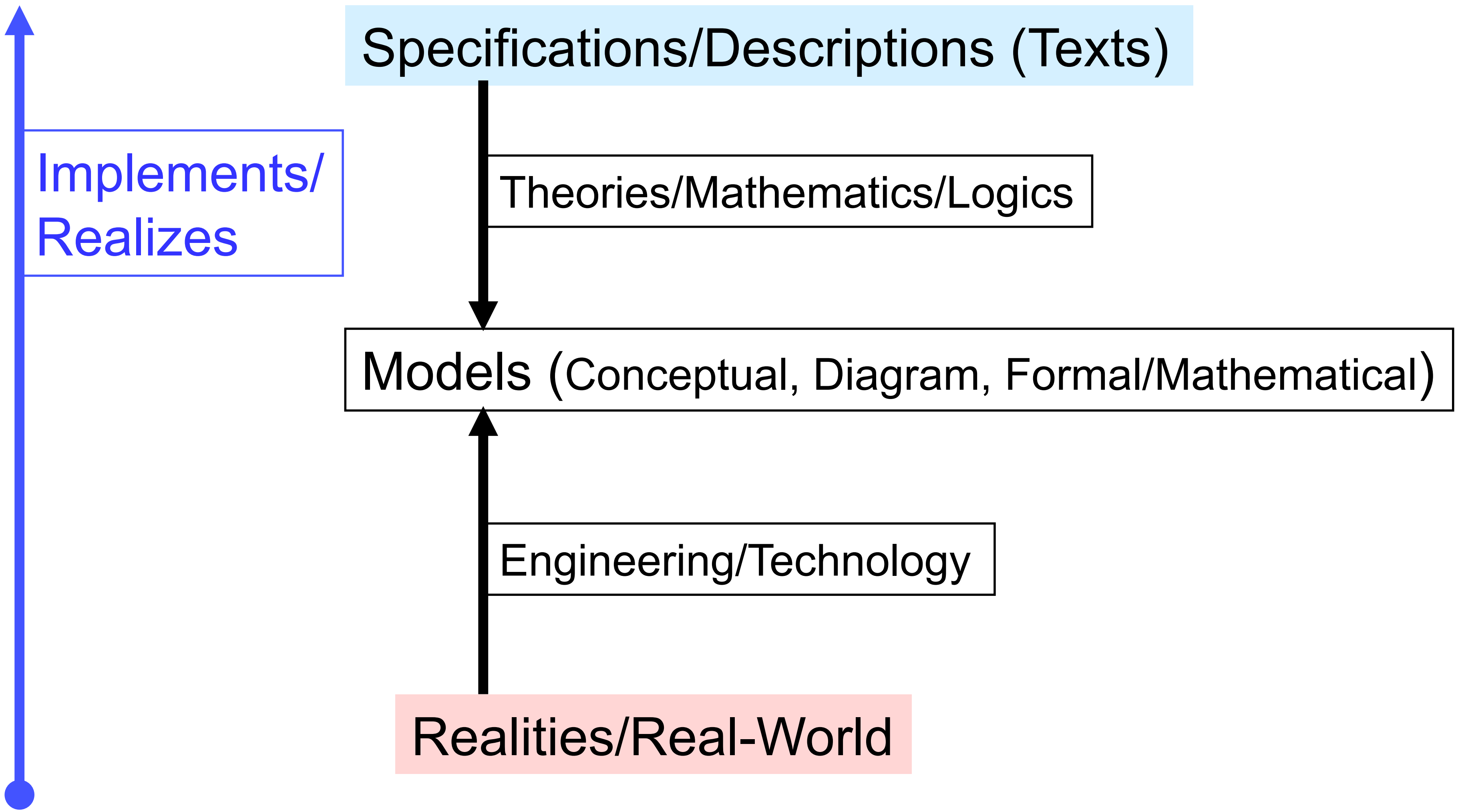
An Overview of Models and Proof Rules for CafeOBJ Proof Scores

FUTATSUGI, Kokichi
二木 厚吉
JAIST

Topics

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

Specifications, Models, Realities



Specification

An **constructor-based equational specification SPEC** in CafeOBJ (a text in the CafeOBJ notation with only equational 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 partially ordered set **S** of sorts, an indexed family **F** of sets of **S**-sorted functions/ operations, and a set **F^c** of constructors. **F^c** is a family of subsets of **F**, i.e. $\mathbf{F^c} \subseteq \mathbf{F}$.

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

Model: (S,F)-Algebra

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

An order-sorted 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 **A_s**and an operation (function) symbol **f :s1 s2 …sn->s(n+1)** in **F** as a function **A_f : A_{s1},A_{s2},..,A_{sn}->$A_{s(n+1)}$.** The interpretation respects the order-sort constrains.

Model: (S,F,F^C)-Algebra

If a sort $s \in S$ is the co-arity of some operator $f \in F^C$, the sort s is called a **constrained sort**. A sort which is not constrained is called a **loose sort**.

An (S,F)-algebra A is called **(S,F,F^C)-algebra** if any value $v \in A_s$ for any constrained sort $s \in S$ is expressible only using

(1) function A_f for $f \in F^C$

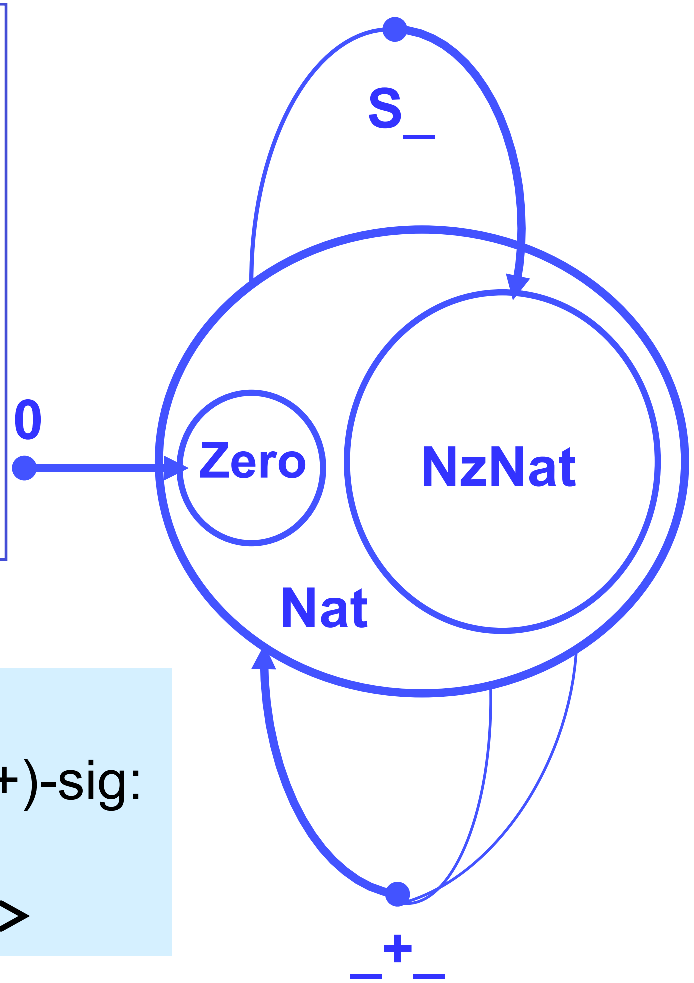
and

(2) function A_g for $g \in F$ whose co-arity is loose sort .

(S,F,F^C)-algebra can also be called **F^C-reachable algebra**

An example of Signature and its Algebra

```
-- Let (PNAT+)-sig be
-- the  signature of PNAT+
-- sort
[ Zero NzNat < Nat ]
-- operators
op 0 :   -> Nat {constr}
op s_ :  Nat -> NzNat {constr}
op _+_ : Nat Nat -> Nat
```



A (PNAT+)-sig-algebra
Order-Sorted Algebra with Signature (PNAT+)-sig:

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

Valuation, evaluation

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 $\mathbf{A_s}$

Equation

Given terms t, t',t1,t1',t2,t2'…tn,tn' , a **conditional equation** is a sentence of the form:

$$\mathbf{t = t' \text{ if } (t1 = t1') \land (t2 = t2') \land \ \ldots \land \ (tn = tn')}$$

An ordinary equation is a sentence of the form:

$$\mathbf{t = t'}$$

that is n=0.

A conditional equation in CafeOBJ notation:

(t=t' if c)

where t,t' are any terms and c is a Boolean term is an abbreviation of

(t=t' if c = true)

Satisfiability of equation

An ordered-sorted algebra **A** **satisfies** a conditional equation:

t = t' if (t1 = t1') /\ (t2 = t2') /\…/\ (tn = tn')

iff

A_v(t1)=A_v(t1') and A_v(t2)=A_v(t2') and…and A_v(tn)=A_v(tn')

implies A_v(t)=A_v(t')

for any valuation **v** .

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

A |= (t = t' if (t1 = t1') /\ (t2 = t2') /\…/\ (tn = tn'))

CafeOBJ _=_ (meta-level equality) and Boolean _=_ (object level equality)

If a specification SP includes,

```
op _=_ : S S -> Bool .
eq (X = X) = true .
ceq X = Y if (X = Y) .
```

then

SP |= t=t' if (t1=t1')/\ (t2=t2')/\.../\(tn=tn')

iff

**SP |= (t1=t1' and t2=t2' and ...and tn=tn'
implies t=t') = true .**

1. **Object-level equality can substitutes for meta-level equality**
2. **Every sentence (conditional equation) can be written as Boolean term.**

SPEC-algebra

For a specification **SPEC** = (**(S,F,F^c), E)**, a **SPEC-algebra** is a **(S,F,F^C)-algebra** which satisfies all equations in **E** .

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** = true) 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 describes some important property of the system which **SPEC** specifies.

Proof rules for
SPEC |= prop (semantic entailment)

For doing formal verification, it is common to think of syntactic (proof theoretic) entailment:

SPEC |- prop

which corresponds to semantic entailment:

SPEC |= prop .

We have a sound and *quasi* complete set of proof rules for |- which satisfies:

SPEC |- prop   iff   SPEC |= prop

for unstructured specifications and constitutes a theoretical foundation for verifications with proof scores.

Proof Rules (1) -- entailment system
(S, P, or Ei denotes a set of equations)

Monotonicity:

$$\frac{S \mid\!- P}{S \cup E1 \mid\!- P \cup E2}$$

(for any E2 ⊆ E1)

Transitivity:

$$\frac{S \cup E1 \mid\!- P \cup E2 \;,\; S \cup E2 \mid\!- P \cup E3}{S \cup E1 \mid\!- P \cup E3}$$

Unions:

$$\frac{S \cup E1 \mid\!- P \cup E2 \;,\; S \cup E1 \mid\!- P \cup E3}{S \cup E1 \mid\!- P \cup (E2 \cup E3)}$$

Translation:

$$\frac{S \mid\!-_{\Sigma} P}{\varphi(S) \mid\!-_{\Sigma'} \varphi(P)}$$

(for any signature morphism $\varphi: \Sigma \text{ -> } \Sigma'$)

Proof Rules (2) -- equational reasoning
(t or ti denotes term and f denotes operator)

Reflexivity:

```
        S |- P
--------------------------
    S |- P U {t=t}
```

Symmetry:

```
                  S |- P
------------------------------------------------
        S U {t1=t2} |- P U {t2=t1}
```

Transitivity:

```
                  S |- P
------------------------------------------------
   S U {t1=t2, t2=t3} |- P U {t1=t3}
```

Congruence:

```
                                   S |- P
------------------------------------------------------------------------------------
 S U {t1=t1',t2=t2',…,tn=tn'} |- P U {f(t1,t2,…,tn)=f(t1',t2',…,tn'}
```

Proof Rules (3)
(H denotes a set of equations, p denotes predicate, X, Y, or Z denotes set of variables, x denotes variable)

Implication:

$$\frac{S \mid\!- P \cup \{(\wedge H \Rightarrow p)\}}{S \cup H \mid\!- P \cup \{p\}} \quad \text{and} \quad \frac{S \cup H \mid\!- P \cup \{p\}}{S \mid\!- P \cup \{(\wedge H \Rightarrow p)\}}$$

Substitutivity:

$$\frac{S \mid\!- P}{S \cup \{(\forall x)p\} \mid\!- P \cup \{(\forall Y)p(x \leftarrow t)\}}$$

Generalization:

$$\frac{S \mid\!-_{\Sigma} P \cup \{(\forall Z)p\}}{S \mid\!-_{\Sigma(Z)} P \cup \{p\}} \quad \text{and} \quad \frac{S \mid\!-_{\Sigma(Z)} P \cup \{p\}}{S \mid\!-_{\Sigma} P \cup \{(\forall Z)p\}}$$

Proof Rules (4)
(H denotes a set of equations, p denotes predicate, X, Y, or Z denotes set of variables, x denotes variable)

C-Abstraction:

$$\frac{\{\, (S \mid\!- P \cup \{(\forall Y)p(x \leftarrow t)\}) \mid t \text{ is constructor } Y\text{-tem, } Y \text{ are loose vars.} \,\}}{S \mid\!- P \cup \{(\forall x)p\}}$$

Case Analysis:

$$\frac{\{\, (S \cup \{f(t1,\dots,tn)=t\} \mid\!-_{\Sigma(Y)} P \cup \{p\}) \mid t \text{ is const. } Y\text{-tem, } Y \text{ are loose vars.} \,\}}{S \mid\!-_{\Sigma} P \cup \{p\}}$$