# Constructor-based Logics

Lecture Note 03b
CafeOBJ Team for JAIST-FSSV2010

# Overview

- extension of CafeOBJ logic to a logic with constructors (in the signatures)
- this logic may be seen as the underlying logic of an (under developing) language

We use:

1. CafeOBJ notation for examples, and
2. CafeOBJ rewriting engine for proofs.

# What we do...

Constructor-based logics = base logic + restriction to reachable models

- define entailment systems for the constructor-based logics;
- investigate soundness, completeness and initiality;

Set the logical foundations for OTS method (FMOODS 2002, Inf Process Lett. 2003, VSTTE 2005);

# Related work

- Equational specification and programming - basis of modern algebraic specification.
- Birkhoff 1935 *On the structure of abstract algebras* - completeness result for equational logic, unsorted case.
- Goguen and Meseguer 1985 *Completeness of many-sorted equational logic* - many-sorted case
- Codescu and Gaina 2008 *Birkhoff Completeness in Institutions* - framework of institutions.
- *Constructor-based Institutions* (present work) .

# General structure of logics

## Framework adopted

The key ingredients of a logic:

- **signatures** and **sentences**,
- **entailment** of a sentence from a set of axioms,

  Entailment systems are represented by its generators = proof rules
- **model** and **satisfaction** of a sentence by a model.

# The Concept of Institution (The semantic part)

An institution (Goguen and Burstall ACM 1992)
$\mathcal{I} = (\mathbb{S}ig, \mathbb{S}en, \mathbb{M}od, \models)$:

- category of **signatures** $\mathbb{S}ig$,
- **sentence** functor $\mathbb{S}en : \mathbb{S}ig \to \mathbb{S}et$ ,
- **model** functor $\mathbb{M}od : \mathbb{S}ig^{op} \to \mathbb{C}at$,
- for each signature $\Sigma$, a **satisfaction relation** $\models_\Sigma$ between $\Sigma$-models and $\Sigma$-sentences s.t. **the satisfaction condition holds**

$$\begin{array}{ccc} \Sigma' & \quad & M' \models \varphi(\rho) \\ \varphi \uparrow & & \Updownarrow \\ \Sigma & & \mathbb{M}od(\varphi)(M') \models \rho \end{array}$$

# First Order Logic with equality (FOL)

- Signatures $(S, F, P)$
  - $S$ -sorts
  - $F$ - function symbols
  - $P$ - predicate symbols
- $(S, F, P)$-models interprets
  - sorts as carrier sets
  - function symbols as functions
  - predicate symbols as relations
- $(S, F, P)$-sentences
  1. two kinds of atoms:
     - equations: $t = t'$
     - relations: $\pi(t_1, \ldots, t_n)$
  2. full sentences: $(\neg, \vee, \mathit{false}, \exists)$atoms.
- The usual Tarskian satisfaction based on the interpretation of terms.

# Horn clause logic (**HCL**)

**Universal Horn sentence** $(\forall X) \bigwedge H \Rightarrow C$

- $X$ finite set of variables
- $H$ finite set of (equational or relational) atoms
- $C$ an atom

**HCL** is the restriction of **FOL** to universal Horn sentences.

# Constructor-based Horn clause logic (**CHCL**) I

- Sign. $(S, F, F^c, P)$ with **constructors** $F^c \subseteq F$
  1. **constrained sorts** $S^c \subseteq S$,
     $(s \in S^c)$ iff (there is $\sigma \in F^c_{w \to s}$)
  2. **loose sorts** $S^l = S - S^c$.

- $(S, F, F^c, P)$-models $M$:
  there exists $f : Y \to M$ (vars $Y$ are of loose sort) s.t.
  $(s \in S^c)$ $f^\#_s : (T_{F^c}(Y))_s \to M_s$ is a surjection

  $\boxed{f^\# : T_{F^c}(Y) \to M \text{ is the unique extension of } f \text{ to a } (S, F^c, P)\text{-morphism.}}$

The models $M$ are reachable (through constructors and loose elements $Y$).

# Constructor-based Horn clause logic (**CHCL**) II

- **Universal Horn sentences** $(\forall X)(\forall Y) \bigwedge H \Rightarrow C$:
  - $X$ - finite set of vars. of constrained sort
  - $Y$ - finite set of vars of loose sort
  - $H$ finite set of atoms, and
  - $C$ an atom

- Sign. morphisms $\varphi : (S, F, F^c, P) \to (S_1, F_1, F_1^c, P_1)$
  1. if $\sigma \in F^c$ then $\varphi(\sigma) \in F_1^c$, and
  2. if $\sigma_1 \in (F_1^c)_{w_1 \to s_1}$, $s_1 \in \varphi(S^c)$ then $\exists \sigma \in F^c$ s. t. $\varphi(\sigma) = \sigma_1$.

- The satisfaction relation is inherited from **FOL**.

# Entailment systems (The syntactic part)

An entailment system $\mathcal{E} = (\mathbb{S}ig, \mathbb{S}en, \vdash)$

*(Monotonicity)* $\dfrac{}{E_1 \vdash E_2}$ whenever $E_2 \subseteq E_1$

*(Transitivity)* $\dfrac{E_1 \vdash E_2, E_2 \vdash E_3}{E_1 \vdash E_3}$

*(Unions)* $\dfrac{E_1 \vdash E_2, E_1 \vdash E_3}{E_1 \vdash E_2 \cup E_3}$

*(Translation)* $\dfrac{E \vdash_\Sigma E'}{\varphi(E) \vdash_{\Sigma'} \varphi(E')}$ for all signature morphisms $\varphi : \Sigma \to \Sigma'$

## Definition (compactness)

$\mathcal{E}$ is compact whenever $\Gamma \vdash \rho$ there exists a finite $\Gamma_f \subseteq \Gamma$ such that $\Gamma_f \vdash \rho$.

# Soundness and Completeness

Logic = $(\mathbb{S}ig, \mathbb{S}en, \mathbb{M}od, \models, \vdash)$
Correctness of proof rules is justified by model theoretic means.

1. sound: $\Gamma \vdash \rho$ implies $\Gamma \models \rho$.
2. complete: $\Gamma \models \rho$ implies $\Gamma \vdash \rho$.

# Entailment System of **CHCL** I

| | |
|---|---|
| AES | *(Reflexivity)* $\dfrac{}{\emptyset \vdash t = t}$ <br> *(Symmetry)* $\dfrac{}{t = t' \vdash t' = t}$ <br> *(Transitivity)* $\dfrac{}{\{t = t', t' = t''\} \vdash t = t''}$ <br> *(Congruence)* $\dfrac{}{\{t_i = t'_i \mid i = \overline{1, n}\} \vdash \sigma(t_1, \ldots, t_n) = \sigma(t'_1, \ldots, t'_n)}$ <br> *(P-Congruence)* $\dfrac{}{\{t_i = t'_i \mid i = \overline{1, n}\} \cup \{\pi(t_1, \ldots, t_n)\} \vdash \pi(t'_1, \ldots, t'_n)}$ |
| IES | *(Implications)* $\dfrac{\Gamma \vdash \bigwedge H \Rightarrow C}{\Gamma \cup H \vdash C}$ and $\dfrac{\Gamma \cup H \vdash C}{\Gamma \vdash \bigwedge H \Rightarrow C}$ |
| GUES | *(Substitutivity)* $\dfrac{}{(\forall x)\rho \vdash (\forall Y)\rho(x \leftarrow t)}$ <br> *(Generalization)* $\dfrac{\Gamma \vdash_{\Sigma} (\forall Z)\rho}{\Gamma \vdash_{\Sigma(Z)} \rho}$ and $\dfrac{\Gamma \vdash_{\Sigma(Z)} \rho}{\Gamma \vdash_{\Sigma} (\forall Z)\rho}$ |

# Entailment System of **CHCL** II

**Theorem (Soundness + Completeness)**

*The restriction of* **CHCL** *to the sentences of the form* $(\forall Y) \bigwedge H \Rightarrow C$*, with* $Y$ *vars. of loose sort, is sound and complete.*

**Notation**

*Let* $\Sigma = (S, F, F^c, P)$ *be a signature.*

- $t$ *is a* $(F \cup Y)$*-term, or for short* *$Y$-term**, where* $Y$ *is a set of vars, if* $t \in T_F(Y)$*;*
- $t$ *is a* *constructor term* *if* $t \in T_{F^c}(Y)$ *and* $Y$ *are vars of loose sort;*

We need rules to deal with universal quantification over variables of constrained sort.

| | |
|---|---|
| RUES | (C-Abstraction) $\dfrac{\{\Gamma \vdash_\Sigma (\forall Y)\rho(x \leftarrow t) \mid Y \text{ are loose vars, } t \text{ is constructor } Y\text{-term}\}}{\Gamma \vdash_\Sigma (\forall x)\rho}$ |

In many cases the premises of the above infinitary rule can be checked using inductive arguments.

# Sufficient completeness

Let $(S, F, F^c, P)$ be a signature; $F^{S^c}$ denotes the set of op. of constrained sort.

## Definition

$\Gamma \subseteq \mathbb{S}en(S, F, F^c, P)$ is sufficient-complete if $\forall t \in T_{F^{S^c}}(Y)$, ($Y$ consists of vars. of loose sort), $\exists t' \in T_{F^c}(Y)$ s.t. $\Gamma \vdash (\forall Y) t = t'$

## Example

```
mod* SP {
[Nat]
op 0 :  -> Nat {constr}
op s_ :  Nat -> Nat {constr}
op _+_ :  Nat Nat -> Nat
vars M N : Nat
eq [lid] :  0 + N = N .
eq [ladd] :  s M + N = s (M + N) . }
```

# Soundness, Completeness and Initiality

**Theorem (Soundness+ quasi-Completeness)**

1. *The entailment system of* **CHCL** *is sound*
2. $\Gamma \vdash_\Sigma \rho$ *if* $\Gamma \models_\Sigma \rho$ *when* $\Gamma$ *is sufficient-complete.*

**Theorem (Initiality)**

*Every sufficient complete set of sentences* $\Gamma$ *has an initial model,* ($\exists M_\Gamma$ *s.t. for all* $M \models \Gamma$ *there exists an unique morphism* $M_\Gamma \to M$).

# Sufficient completeness assumption

**Example**

```
mod* SPEC {
[S]
- constructors
op a :  -> S {constr}
- operators
op b :  -> S }
```

1. Completeness:
   $\emptyset \models$ `a = b` but $\emptyset \not\vdash$ `a = b` because `SPEC` is not sufficient complete.
2. Initiality:
   - $\mathbb{N}$ is initial model of `SP`.
   - `SP` without `ladd` does not have initial model.

Initiality(Sufficient completeness) is not needed to reason about inductive properties.

# Induction Scheme I

We want `SP` $\vdash$ `(∀x) (∀y) x + s y = s(x + y)`. By *C-Abstraction* we need

1. `SP` $\vdash$ `(∀y) 0 + s y = s (0 + y)`
2. `SP` $\vdash$ `(∀y) s 0 + s y = s (s 0 + y)`
3. `SP` $\vdash$ `(∀y) s s 0 + s y = s (s s 0 + y)`

⋮

It is required an inductive argument:

- IB `SP` $\vdash$ `(∀y) 0 + s y = s (0 + y)`
- IS `SP` $\cup$ { `(∀y) a + s y = s(a+y)` } $\vdash$ `(∀y) s a + s y = s(s a + y)`

$$\dfrac{\dfrac{\overset{\text{CafeOBJ code}}{\overline{\texttt{SP}\vdash\texttt{(∀y) 0+sy=s(0+y)}}} \qquad \overset{\text{CafeOBJ code}}{\overline{\texttt{SP}\cup\{\texttt{(∀y) a+sy=s(a+y)}\}\vdash\texttt{(∀y) sa+sy=s(sa+y)}}}}{\texttt{SP}\vdash\texttt{(∀y) 0+sy=s(0+y)},\ \texttt{SP}\vdash\texttt{(∀y) s0+sy=s(s0+y)},\ \ldots}}{\texttt{SP}\vdash\texttt{(∀x) (∀y) x + s y = s(x + y)}}$$

# Induction Scheme II

CafeOBJ code:

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

IS
```
open SP
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]
op _=_ :  Elt Elt -> Bool
vars X Y : Elt
eq [equal] :  (X = X) = true .
ceq [cequal] :  X = Y if (X = Y) . }
```

## Lemma (Equality)

1. {equal, cequal,a=b} $\vdash_{\texttt{SPEC(a,b)}}$ (a=b)=true
2. {equal, cequal,(a=b)=true} $\vdash_{\texttt{SPEC(a,b)}}$ a=b
3. {equal, cequal, true=false} $\vdash_{\texttt{SPEC}}$ $(\forall$x$)$ $(\forall$y$)$ x=y

# Case Analysis I

- $(\Sigma, E)$, specification with $\Sigma = (S, F, F^c)$.
- $\sigma \in (F_{s_1 \ldots s_n \to s} - F^c_{s_1 \ldots s_n \to s})$ operation of constrained sort $s$
- $t_1, \ldots, t_n$ constructor terms
- $\sigma(t_1, \ldots, t_n)$ is "not defined", i.e. $(\nexists)$ constructor term $t$ such that $E \vdash_{\Sigma(Y)} \sigma(t_1, \ldots, t_n) = t$, where $Y$ are all the variables in $t$ and $\sigma(t_1, \ldots, t_n)$

$$\textit{(Case Analysis)}\ \frac{\{\Gamma \cup \{\sigma(t_1, \ldots, t_n) = t\} \vdash_{\Sigma(Y)} e \mid \ Y \text{ are loose vars, } t \text{ is constructor } Y\text{-term}\}}{\Gamma \vdash_{\Sigma} e}$$

To prove `SPEC⊢a=b` by *Case Analysis* we need `SPEC∪{a=b}⊢a=b` which is obvious

# Case Analysis II

**Remark**

The set of terms $t$ above may be infinite and therefore premises of *Case Analysis* may be infinite too. But the sort $s$ may have
- one constructor such as the sort `S` of `SPEC` (there is one constructor `a`), or
- two constructors such as the sort `Bool` (there are two constructors `true` and `false`,
and the premises of *Case Analysis* are finite.

The cases to analyze, after applying *Case Analysis* rule, are sufficient complete; therefore for any sematic consequence $\Gamma \models \rho$ there is a an entailment $\Gamma \vdash \rho$.

# Entailment System of **CHCL** I

| | |
|---|---|
| AES | *(Reflexivity)* $\dfrac{}{\emptyset \vdash t = t}$<br>*(Symmetry)* $\dfrac{}{t = t' \vdash t' = t}$<br>*(Transitivity)* $\dfrac{}{\{t = t', t' = t''\} \vdash t = t''}$<br>*(Congruence)* $\dfrac{}{\{t_i = t'_i \mid i = \overline{1,n}\} \vdash \sigma(t_1, ..., t_n) = \sigma(t'_1, ..., t'_n)}$<br>*(P-Congruence)* $\dfrac{}{\{t_i = t'_i \mid i = \overline{1,n}\} \cup \{\pi(t_1, ..., t_n)\} \vdash \pi(t'_1, ..., t'_n)}$ |
| IES | *(Implications)* $\dfrac{\Gamma \vdash \bigwedge H \Rightarrow C}{\Gamma \cup H \vdash C}$ and $\dfrac{\Gamma \cup H \vdash C}{\Gamma \vdash \bigwedge H \Rightarrow C}$ |

# Entailment System of **CHCL** II

| | |
|---|---|
| GUES | *(Substitutivity)* $\dfrac{}{(\forall x)\rho \vdash (\forall Y)\rho(x \leftarrow t)}$ <br> *(Generalization)* $\dfrac{\Gamma \vdash_{\Sigma} (\forall Z)\rho}{\Gamma \vdash_{\Sigma(Z)} \rho}$ and $\dfrac{\Gamma \vdash_{\Sigma(Z)} \rho}{\Gamma \vdash_{\Sigma} (\forall Z)\rho}$ |
| RUES | *(C-Abstraction)* $\dfrac{\{\Gamma \vdash_{\Sigma} (\forall Y)\rho(x \leftarrow t) \mid Y \text{ are loose vars, } t \text{ is constructor } Y\text{-term}\}}{\Gamma \vdash_{\Sigma} (\forall x)\rho}$ <br> *(Case Analysis)* $\dfrac{\{\Gamma \cup \{\sigma(t_1, \ldots, t_n) = t\} \vdash_{\Sigma(Y)} e \mid Y \text{ are loose vars, } t \text{ is constructor } Y\text{-term}\}}{\Gamma \vdash_{\Sigma} e}$ |

**Theorem (Soundness+quasi-Completeness)**

1. *The entailment system of* **CHCL** *is sound*
2. $\Gamma \vdash_{\Sigma} \rho$ *if* $\Gamma \models_{\Sigma} \rho$ *when* $\Gamma$ *is sufficient-complete.*

# Gödel Incompleteness

- $\Sigma = (S, F, F^c, P)$ and $F = F^c$ $(S^l = \emptyset)$
- $S^l = \emptyset$ implies all $\Sigma$-models consist of interpretations of terms
- $\Gamma$ an arbitrary set of $\Sigma$-sentences
- $O_\Gamma \to M$ is a surjection for all $\Sigma$-models $M$
- surjective morphisms preserve satisfaction of equations:

$$\Gamma \models (\forall X)t = t' \text{ iff } O_\Gamma \models (\forall X)t = t'$$

- we obtained complete entailment relations to reason about logical consequences of initial models
- Gödel incompleteness theorem: the semantic consequences of specifications in **CHCL** are not recursively enumerable

# Summary

| |
|---|
| Constructor-based institutions = base institution + restriction to reachable models |

Abstract characterization of the concept of reachable model + application to concrete institutions.

- institution-dependent:

  - proof rules for the atomic sentences of each institution

  - soundness and completeness

- institution-independent:

  - assume an entailment system for the 'atomic' part of the institution

  - define the entailment systems in the above figure, abstractly.

  - soundness and the completeness + instantiating the results to **CHCL**, **CHOSA**, **CHPOA**, **CHPA**.

# Future Work

- we are planning to apply the present results to other institutions such as higher-order logic, and to extend the framework possible to modal logics
- extend the framework by adding also observations (behavioral)
- investigate the properties needed to reason about the logical consequences of structured specifications such as amalgamation and interpolation
- specify and verify software system using the developed theoretical framework.