A Sketch of Specification Calculus in CafeOBJ

FUTATSUGI,Kokichi
二木 厚吉
JAIST

Specification Calculus

Let S_i be a specification and p_i be a predicate in S_i, then "$S_i \vdash p_i$" is called a goal and means that "p_i is provable in S_i". In CafeOBJ a specification is denoted by a module.

Proof Goal: $\{S_g \vdash p_g\}$

Proof Rules: $\dfrac{S_{i_1} \vdash p_{i_1}, S_{i_2} \vdash p_{i_2}, \cdots, S_{i_{k(i)}} \vdash p_{i_{k(i)}}}{S_{i_0} \vdash p_{i_0}}$ $(i \in \{1, 2, \cdots, m\})$

Calculation Rules:

- "$S_{i_0} \vdash p_{i_0}$" is replaced with "$S_{i_1} \vdash p_{i_1}, S_{i_2} \vdash p_{i_2}, \cdots S_{i_{k(i)}} \vdash p_{i_{k(i)}}$".
- "$S_j \vdash p_j$" is erased if "$p_j \rightarrow^*_{E_{S_j}}$ `true`".
  That is, if "`red in` S_j `:` p_j `.`" returns `true`.
- `The proof is over if the proof goal becomes empty.`

```
:ctf[ <BooleanTerm> .]
:ctf[tl = tr .]
:csp{<e1> .  <e2> .  <e3> .}
```

CafeOBJ **code**

```
:def <RuleName> = :ctf[ <bt> .]
-- <bt> : Boolean Term
```

Proof Rule `<RuleName>`

$$\frac{S \cup \{\texttt{eq <bt>= true .}\} \vdash p, S \cup \{\texttt{eq <bt>= false .}\} \vdash p}{S \vdash p}$$

```
:ctf[ <BooleanTerm> .]
:ctf[tl = tr .]
:csp{<e1> .  <e2> .  <e3> .}
```

CafeOBJ **Code**

```
:def <RuleName> = :ctf[<l> = <r> .]
```

Proof Rule `<RuleName>`

$$\frac{S \cup \{\texttt{eq <l>=<r> .}\} \vdash p, S \cup \{\texttt{eq (<l>=<r>)} = \texttt{false .}\} \vdash p}{S \vdash p}$$

```
:ctf[ <BooleanTerm> .]
:ctf[tl = tr .]
:csp{<e1> .  <e2> .  <e3> .}
```

CafeOBJ **code**

```
:def <RuleName> = :csp{<e1> <e2> ... <em>}
--  <ei> = eq <til> = <tir> .
```

Proof Rule `<RuleName>`

$$\frac{S \cup \{\texttt{<e1>}\} \vdash p, S \cup \{\texttt{<e2>}\} \vdash p, \cdots, S \cup \{\texttt{<em>}\} \vdash p}{S \vdash p}$$

Hence,

```
:ctf[ <bt> .]
 is equal to
:csp{eq <bt> = true . eq <bt> = false .}
```

```
-- check the step
select SET^in-iStep .
:goal{eq iStep = true .}
:def csp-1 = :csp{eq e1 = e2 . eq (e1 =t e2) = false .}
:def ctf-2 = :ctf[e2 in s2 .]
:apply(csp-1 rd- ctf-2 rd-)
:show proof
```

```
:goal{eq iStep = true .}
 =outPut=>
:goal { ** root ------------------------------------
  -- context module: SET^in-iStep
  -- sentence to be proved
    eq iStep = true .
}
** Initial goal (root) is generated. **
```

Proof Score (1)

```
:goal{eq iStep = true .}
:apply(csp-1 rd- ctf-2 rd-)(1)
:apply(ctf-1 rd- ctf-2 rd-)(2)
```

```
:show proof
 =outPut=>
root*
[csp-1] 1*
[ctf-2] 1-1*
[ctf-2] 1-2*
[csp-1] 2*
[ctf-2] 2-1*
[ctf-2] 2-2*
      | |ctf-2
      |csp-1
```

Proof Score (1)

```
:goal{eq iStep = true .}
:apply(csp-1 rd- ctf-2 rd-)(1)
:apply(ctf-1 rd- ctf-2 rd-)(2)
```

```
root*
[csp-1] 1*
[ctf-2] 1-1*
[ctf-2] 1-2*
[csp-1] 2*
[ctf-2] 2-1*
[ctf-2] 2-2*
      | |csp-2
      |ctf-1
```

```
  csp-1 -> root
          /   \
ctf-2 -> 1     2
        / \   / \
       1   2 1   2
```

```
:apply(csp-1 rd- ctf-2 rd-)
 =isEqualto=>
:apply(csp-1 rd- ctf-2 rd-)
:apply(ctf-2 rd-)
:apply(ctf-2 rd-)
```

```
SET^in-iStep(X.SET^)> :apply(csp-1 rd-)
...
SET^in-iStep(X.SET^)> :show proof
root
>[csp-1] 1
[csp-1] 2

SET^in-iStep(X.SET^)> :apply(ctf-2 rd-)
...
SET^in-iStep(X.SET^)> :show proof
root
[csp-1] 1*
[ctf-2] 1-1*
[ctf-2] 1-2*
>[csp-1] 2

SET^in-iStep(X.SET^)> :apply(ctf-2 rd-)
...
SET^in-iStep(X.SET^)> :show proof
root*
[csp-1] 1*
[ctf-2] 1-1*
[ctf-2] 1-2*
[csp-1] 2*
[ctf-2] 2-1*
[ctf-2] 2-2*
```