Appendices to A Module System for Domain-Specific Languages

ETHAN K. JACKSON
Research In Software Engineering (RiSE)
Microsoft Research, Redmond, WA, USA 98052
(*e-mail:* ejackson@microsoft.com)

submitted ; revised ; accepted

Appendix A Action Language

```
1:  domain Actions
2:  {
3:      //// Declarations
4:      VarDecl ::= fun (id: String -> type: { BOOL, INT }).
5:      ActDecl ::= fun (id: String -> action: any Action).
6:
7:      //// Action language (expressions)
8:      BoolOp ::= { NOT, AND, OR }.
9:      IntOp  ::= { NEG, ADD, SUB, MUL, DIV }.
10:     CmpOp  ::= { LT, LE, GT, GE, EQ, NEQ }.
11:
12:     Var   ::= new (id: String).
13:     UnApp ::= new (op: { NEG, NOT }, arg1: any Expr).
14:     BnApp ::= new (op: { ADD, SUB, MUL, DIV, AND, OR } + CmpOp,
15:                    arg1: any Expr, arg2: any Expr).
16:     Expr  ::= Var + UnApp + BnApp + Boolean + Integer.
17:
18:     //// Action language (statements)
19:     Asn ::= new (var: String, expr: any Expr).
20:     ITE ::= new (cond: any Expr, true: any Action, false: any Action).
21:     Seq ::= new (act1: any Action, act2: any Action).
22:     Action ::= Asn + ITE + Seq + { NOP }.
23:
24:     //// Static typing
25:     Sub ::= (Action + Expr).
26:     Sub(e)                    :- ActDecl(_, e);
27:                                  Sub(UnApp(_, e));
28:                                  Sub(Asn(_, e)).
29:     Sub(e), Sub(e')           :- Sub(BnApp(_, e, e'));
30:                                  Sub(Seq(e, e')).
31:     Sub(e), Sub(e'), Sub(e'') :- Sub(ITE(e, e', e'')).
32:
33:     TypeJudge ::= (Action + Expr, { BOOL, INT, ANY }).
34:
```

```
35:    TypeJudge(e, INT) :-
36:       Sub(e), e : Integer;
37:       Sub(e), e = Var(n), VarDecl(n, INT);
38:       Sub(e), e = UnApp(op, e'), op : IntOp, TypeJudge(e', INT);
39:       Sub(e), e = Asn(n, e'), VarDecl(n, INT), TypeJudge(e', INT);
40:       Sub(e), e = BnApp(op, e', e''), op : IntOp,
41:          TypeJudge(e', INT), TypeJudge(e'', INT).
42:
43:    TypeJudge(e, BOOL) :-
44:       Sub(e), e : Boolean;
45:       Sub(e), e = Var(n), VarDecl(n, BOOL);
46:       Sub(e), e = UnApp(op, e'), op : BoolOp, TypeJudge(e', BOOL);
47:       Sub(e), e = Asn(n, e'), VarDecl(n, BOOL), TypeJudge(e', BOOL).
48:       Sub(e), e = BnApp(op, e', e''), op : BoolOp,
49:          TypeJudge(e', BOOL), TypeJudge(e'', BOOL);
50:       Sub(e), e = BnApp(op, e', e''), op : CmpOp,
51:          TypeJudge(e', t), TypeJudge(e'', t).
52:
53:    TypeJudge(e, ANY) :-
54:       Sub(e), e = NOP;
55:       Sub(e), e = Seq(e', e''), TypeJudge(e', _), TypeJudge(e'', _);
56:       Sub(e), e = ITE(e', e'', e'''), TypeJudge(e', BOOL),
57:          TypeJudge(e'', _), TypeJudge(e''', _).
58:
59:    conforms no { e | Sub(e), no { t | TypeJudge(e, t) } }.
60: }
```

Appendix B Example Symbol Table

Qualifiers	Name	Kind, Arity	Qualifiers	Name	Kind, Arity
	ADD	η, 0	left	Init	η, 1
	AND	η, 0	left	IntOp	μ, 0
	ANY	η, 0	left	Reach	δ, 1
	BOOL	η, 0	left	Seq	η, 2
	DIV	η, 0	left	State	η, 1
	EQ	η, 0	left	Sub	δ, 1
	FALSE	η, 0	left	Trans	η, 3
	GE	η, 0	left	TypeJudge	δ, 2
	GT	η, 0	left	UnApp	η, 2
	INT	η, 0	left	Var	η, 1
	LE	η, 0	left	VarDecl	η, 2
	LT	η, 0	left.Actions	conforms	δ, 0
	MUL	η, 0	left.DetFSMWithActions	conforms	δ, 0
	NEG	η, 0	left.MachTwoState	eFoo	σ, 0
	NEQ	η, 0	left.MachTwoState	s1	σ, 0
	NOP	η, 0	left.MachTwoState	s2	σ, 0
	NOT	η, 0	left.NonDetFSM	conforms	δ, 0
	OR	η, 0	right	ActDecl	η, 2
	SUB	η, 0	right	ActMap	η, 2
	TRUE	η, 0	right	Action	μ, 0
	a	ν, 0	right	Asn	η, 2
	e	ν, 0	right	BnApp	η, 3
	e’	ν, 0	right	BoolOp	μ, 0
	e’’	ν, 0	right	CmpOp	μ, 0
	e’’’	ν, 0	right	Event	η, 1
	i	ν, 0	right	Expr	μ, 0
	n	ν, 0	right	ITE	η, 3
	op	ν, 0	right	Init	η, 1
	s	ν, 0	right	IntOp	μ, 0
	s’	ν, 0	right	Reach	δ, 1
	s’’	ν, 0	right	Seq	η, 2
	t	ν, 0	right	State	η, 1
ParallelFSMs	conforms	δ, 0	right	Sub	δ, 1
left	ActDecl	η, 2	right	Trans	η, 3
left	ActMap	η, 2	right	TypeJudge	δ, 2
left	Action	μ, 2	right	UnApp	η, 2
left	Asn	η, 2	right	Var	η, 1
left	BnApp	η, 3	right	VarDecl	η, 2
left	BoolOp	μ, 0	right.Actions	conforms	δ, 0
left	CmpOp	μ, 0	right.DetFSMWithActions	conforms	δ, 0
left	Event	η, 1	right.MachTwoState	eFoo	σ, 0
left	Expr	μ, 0	right.MachTwoState	s1	σ, 0
left	ITE	η, 3	right.MachTwoState	s2	σ, 0
			right.NonDetFSM	conforms	δ, 0

Table B 1. Symbol table of composite model *ParallelCntrs* in Figure 5.