1: Operators of Differential Dynamic Logic (dL)

dL	KeYmaera X	Operator	Meaning
$e = d$	`e=d`	equals	true if values of terms e and d are equal
$e \geq d$	`e>=d`	greater-or-equal	true if value of e greater-or-equal to value of d
$p(e_1, \ldots, e_k)$	`p(e1,...,ek)`	predicate	true if p holds for the value of $(e1, \ldots, ek)$
$\neg P$	`!P`	not / negation	true if P is false
$P \wedge Q$	`P & Q`	and / conjunction	true if both P and Q are true
$P \vee Q$	`P | Q`	or / disjunction	true if P is true or if Q is true
$P \rightarrow Q$	`P -> Q`	implies / implication	true if P is false or Q is true
$P \leftrightarrow Q$	`P <-> Q`	equivalent / bi-implication	true if P and Q are both true or both false
$\forall x\, P$	`\forall x P`	for all / universal quantifier	true if P is true for all real values of variable x
$\exists x\, P$	`\exists x P`	exists / existential quantifier	true if P is true for some real value of variable x
$[a]P$	`[a]P`	box / $[\cdot]$ modality	true if P is true after all runs of HP a
$\langle a \rangle P$	`<a>P`	diamond / $\langle\cdot\rangle$ modality	true if P is true after at least one run of HP a

Unary operators (including $\forall x, \exists x, [\alpha], \langle\alpha\rangle$) bind stronger than binary operators. And ; binds stronger than $\cup$.

2: Statements and effects of Hybrid Programs (HPs)

HP	KeYmaera X	Operation	Effect
$x := e$	`x:=e;`	discrete assignment	assigns value of term e to variable x
$x := *$	`x:=*;`	nondeterministic assign	assigns any real value to variable x
$x' = f(x)\,\&\,Q$	`{x'=f(x) & Q}`	continuous evolution	evolve along differential equation $x' = f(x)$ within evolution domain Q for any duration
$?Q$	`?Q;`	test	check first-order formula Q at current state
$a; b$	`a b`	sequential composition	HP b starts after HP a finishes
$a \cup b$	`a ++ b`	nondeterministic choice	choice between alternatives HP a or HP b
a^*	`{a}*`	nondeterministic repetition	repeats HP a n-times for any $n \in \mathbb{N}$

```
Definitions          /* function symbols cannot change their value */
    Real A = 5;      /* real-valued maximum acceleration constant is defined as 5 */
    Real B;          /* real-valued maximum braking constant is arbitrary */
End.

ProgramVariables     /* program variables may change their value over time */
    Real x;          /* real-valued position */
    Real v;          /* real-valued velocity */
    Real a;          /* current acceleration chosen by controller */
End.

Problem                                          /* conjecture in differential dynamic logic */
    v>=0 & A>0 & B>0                             /* initial condition */
  ->                                             /* implies */
  [                                              /* all runs of hybrid program in [...] */
    {                                            /* braces {} for grouping of programs */
      {?v<=5;a:=A; ++ a:=0; ++ a:=-B;}           /* nondeterministic choice of acceleration a */
      {x'=v, v'=a & v>=0}                        /* differential equation system with domain */
    }* @invariant(v>=0)                          /* loop repeats, @invariant contract */
  ] v>=0                                         /* safety/postcondition after hybrid program */
End.
```

3: Axioms

assignb [:=] $[x := e]p(x) \leftrightarrow p(e)$

randomb [:*] $[x := *]p(x) \leftrightarrow \forall x\, p(x)$

testb [?] $[?Q]P \leftrightarrow (Q \rightarrow P)$

solve [′] $[x' = f(x)\ \&\ q(x)]p(x) \leftrightarrow \forall t{\geq}0\ \big((\forall 0{\leq}s{\leq}t\, q(x(s))) \rightarrow [x := x(t)]p(x)\big)$ (if $x'(t) = f(x(t))$)

choiceb [∪] $[a \cup b]P \leftrightarrow [a]P \wedge [b]P$

composeb [;] $[a; b]P \leftrightarrow [a][b]P$

iterateb [*] $[a^*]P \leftrightarrow P \wedge [a][a^*]P$

diamond $\langle\cdot\rangle$ $\neg[a]\neg P \leftrightarrow \langle a\rangle P$

K K $[a](P \rightarrow Q) \rightarrow ([a]P \rightarrow [a]Q)$

I I $[a^*]P \leftrightarrow P \wedge [a^*](P \rightarrow [a]P)$

V V $p \rightarrow [a]p$

4: Differential equation sequent calculus proof rules

dW dW $\dfrac{\Gamma_{\text{const}}, Q{\rightarrow}p(x), \Delta_{\text{const}}}{\Gamma{\rightarrow}[x' = f(x)\,\&\,Q]p(x), \Delta}$

dI dI $\dfrac{\Gamma, Q{\rightarrow}P, \Delta \quad Q{\rightarrow}[x' := f(x)](P)'}{\Gamma{\rightarrow}[x' = f(x)\,\&\,Q]P, \Delta}$

dC dC $\dfrac{\Gamma{\rightarrow}[x' = f(x)\,\&\,Q]C, \Delta \quad \Gamma{\rightarrow}[x' = f(x)\,\&\,Q \wedge C]P, \Delta}{\Gamma{\rightarrow}[x' = f(x)\,\&\,Q]P, \Delta}$

dG dG $\dfrac{\Gamma{\rightarrow}\exists y\,[x' = f(x), y' = a(x)y + b(x)\,\&\,Q]P, \Delta}{\Gamma{\rightarrow}[x' = f(x)\,\&\,Q]P, \Delta}$

5: Propositional sequent calculus proof rules

notR ¬R $\dfrac{\Gamma, P{\rightarrow}\Delta}{\Gamma{\rightarrow}\neg P, \Delta}$ andR ∧R $\dfrac{\Gamma{\rightarrow}P, \Delta \quad \Gamma{\rightarrow}Q, \Delta}{\Gamma{\rightarrow}P \wedge Q, \Delta}$ orR ∨R $\dfrac{\Gamma{\rightarrow}\Delta, P, Q}{\Gamma{\rightarrow}P \vee Q, \Delta}$

notL ¬L $\dfrac{\Gamma{\rightarrow}\Delta, P}{\neg P, \Gamma{\rightarrow}\Delta}$ andL ∧L $\dfrac{\Gamma, P, Q{\rightarrow}\Delta}{P \wedge Q, \Gamma{\rightarrow}\Delta}$ orL ∨L $\dfrac{P, \Gamma{\rightarrow}\Delta \quad Q, \Gamma{\rightarrow}\Delta}{P \vee Q, \Gamma{\rightarrow}\Delta}$

implyR →R $\dfrac{\Gamma, P{\rightarrow}\Delta, Q}{\Gamma{\rightarrow}P \rightarrow Q, \Delta}$ equivR ↔R $\dfrac{\Gamma, P{\rightarrow}\Delta, Q \quad \Gamma, Q{\rightarrow}\Delta, P}{\Gamma{\rightarrow}P \leftrightarrow Q, \Delta}$

implyL →L $\dfrac{\Gamma{\rightarrow}\Delta, P \quad Q, \Gamma{\rightarrow}\Delta}{P \rightarrow Q, \Gamma{\rightarrow}\Delta}$ equivL ↔L $\dfrac{P \wedge Q, \Gamma{\rightarrow}\Delta \quad \neg P \wedge \neg Q, \Gamma{\rightarrow}\Delta}{P \leftrightarrow Q, \Gamma{\rightarrow}\Delta}$

id id $\dfrac{}{P, \Gamma{\rightarrow}P, \Delta}$ closeTrue ⊤R $\dfrac{}{\Gamma{\rightarrow}\mathit{true}, \Delta}$ hideR WR $\dfrac{\Gamma{\rightarrow}\Delta}{\Gamma{\rightarrow}P, \Delta}$ PR $\dfrac{\Gamma{\rightarrow}Q, P, \Delta}{\Gamma{\rightarrow}P, Q, \Delta}$

cut cut $\dfrac{\Gamma, C{\rightarrow}\Delta \quad \Gamma{\rightarrow}\Delta, C}{\Gamma{\rightarrow}\Delta}$ closeFalse ⊥L $\dfrac{}{\mathit{false}, \Gamma{\rightarrow}\Delta}$ hideL WL $\dfrac{\Gamma{\rightarrow}\Delta}{P, \Gamma{\rightarrow}\Delta}$ PL $\dfrac{Q, P, \Gamma{\rightarrow}\Delta}{P, Q, \Gamma{\rightarrow}\Delta}$

6: Quantifier sequent calculus proof rules

allR ∀R $\dfrac{\Gamma{\rightarrow}p(y), \Delta}{\Gamma{\rightarrow}\forall x\, p(x), \Delta}$ $(y \notin \Gamma, \Delta, \forall x\, p(x))$ existsR ∃R $\dfrac{\Gamma{\rightarrow}p(e), \Delta}{\Gamma{\rightarrow}\exists x\, p(x), \Delta}$ (arbitrary term e)

allL ∀L $\dfrac{\Gamma, p(e){\rightarrow}\Delta}{\Gamma, \forall x\, p(x){\rightarrow}\Delta}$ (arbitrary term e) existsL ∃L $\dfrac{\Gamma, p(y){\rightarrow}\Delta}{\Gamma, \exists x\, p(x){\rightarrow}\Delta}$ $(y \notin \Gamma, \Delta, \exists x\, p(x))$

7: dL Sequent calculus proof rules

loop loop $\dfrac{\Gamma \vdash J, \Delta \quad J \vdash P \quad J \vdash [a]J}{\Gamma \vdash [a^*]P, \Delta}$

generalize MR $\dfrac{\Gamma \vdash [a]Q, \Delta \quad Q \vdash P}{\Gamma \vdash [a]P, \Delta}$

generalize ML $\dfrac{\Gamma, [a]Q \vdash \Delta \quad P \vdash Q}{\Gamma, [a]P \vdash \Delta}$

abstract GVR $\dfrac{\Gamma_{\text{const}} \vdash P, \Delta_{\text{const}}}{\Gamma \vdash [a]P, \Delta}$

discreteGhost iG $\dfrac{\Gamma \vdash [y := e]p, \Delta}{\Gamma \vdash p, \Delta}$ (y new)

CEat CER $\dfrac{\Gamma \vdash C(Q), \Delta \quad Q \leftrightarrow P}{\Gamma \vdash C(P), \Delta}$

CEat CEL $\dfrac{\Gamma, C(Q) \vdash \Delta \quad Q \leftrightarrow P}{\Gamma, C(P) \vdash \Delta}$

CEat CQR $\dfrac{\Gamma \vdash p(k), \Delta \quad k = e}{\Gamma \vdash p(e), \Delta}$

CEat CQL $\dfrac{\Gamma, p(k) \vdash \Delta \quad k = e}{\Gamma, p(e) \vdash \Delta}$

CTR $\dfrac{e = k}{\Gamma \vdash c(e) = c(k), \Delta}$

CTL $\dfrac{e = k}{\Gamma, c(e) = c(k) \vdash \Delta}$

US US $\dfrac{\Gamma \vdash \Delta}{\sigma(\Gamma) \vdash \sigma(\Delta)}$ UR $\dfrac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta}$ BRR $\dfrac{\Gamma \vdash [y := e]\varphi, \Delta}{\Gamma \vdash [x := e]\varphi, \Delta}$ BRL $\dfrac{\Gamma, [y := e]\varphi \vdash \Delta}{\Gamma, [x := e]\varphi \vdash \Delta}$ $(y, y', x' \notin \mathrm{FV}(\varphi))$

8: Differential equation axioms

DW DW $[\boldsymbol{x' = f(x) \,\&\, Q}]\boldsymbol{P} \leftrightarrow [x' = f(x) \,\&\, Q](Q \to P)$

DI DI $\big([\boldsymbol{x' = f(x) \,\&\, Q}]\boldsymbol{P} \leftrightarrow [?Q]P\big) \leftarrow (Q \to [x' = f(x) \,\&\, Q])(P)'$)

DC DC $\big([\boldsymbol{x' = f(x) \,\&\, Q}]\boldsymbol{P} \leftrightarrow [x' = f(x) \,\&\, Q \wedge C]P\big) \leftarrow [x' = f(x) \,\&\, Q]C$

DE DE $[\boldsymbol{x' = f(x) \,\&\, Q}]\boldsymbol{P} \leftrightarrow [x' = f(x) \,\&\, Q][x' := f(x)]P$

DG DG $[\boldsymbol{x' = f(x) \,\&\, Q}]\boldsymbol{P} \leftrightarrow \exists y\, [x' = f(x), y' = a(x)y + b(x) \,\&\, Q]P$

9: First-order axioms

allInst ∀i $(\forall x\, p(x)) \to p(e)$

allDist ∀→ $\forall x\, (P \to Q) \to (\forall x\, P \to \forall \boldsymbol{x}\, \boldsymbol{Q})$

allV $\mathrm{V}_\forall$ $p \to \forall \boldsymbol{x}\, \boldsymbol{p}$

∃ $\neg\forall x\, \neg P \leftrightarrow \exists x\, P$

10: Derived rules

cohideR WR $\dfrac{\vdash P}{\Gamma \vdash P, \Delta}$

cohideL WL $\dfrac{P \vdash}{P, \Gamma \vdash \Delta}$

cohide2 WLR $\dfrac{P \vdash Q}{P, \Gamma \vdash Q, \Delta}$

cutR cutR $\dfrac{\Gamma \vdash Q, \Delta \quad \Gamma \vdash Q \to P, \Delta}{\Gamma \vdash P, \Delta}$

cutL cutL $\dfrac{Q, \Gamma \vdash \Delta \quad \Gamma \vdash \Delta, P \to Q}{P, \Gamma \vdash \Delta}$

commuteEquivR $\leftrightarrow c\mathrm{R}$ $\dfrac{\Gamma \vdash Q \leftrightarrow P, \Delta}{\Gamma \vdash P \leftrightarrow Q, \Delta}$

commuteEquivL $\leftrightarrow c\mathrm{L}$ $\dfrac{Q \leftrightarrow P, \Gamma \vdash \Delta}{P \leftrightarrow Q, \Gamma \vdash \Delta}$

equivifyR $\to 2\leftrightarrow$ $\dfrac{\Gamma \vdash P \leftrightarrow Q, \Delta}{\Gamma \vdash P \to Q, \Delta}$

11: Differential axioms

DS	DS	$[\boldsymbol{x' = c() \,\&\, q(x)}]\boldsymbol{p(x)} \leftrightarrow \forall t{\geq}0\, \big((\forall 0{\leq}s{\leq}t\, q(x+c()s)) \rightarrow [x := x + c()t]p(x)\big)$
Dconst	c'	$\boldsymbol{(c())'} = 0$
Dvar	x'	$\boldsymbol{(x)'} = x'$
Dplus	$+'$	$\boldsymbol{(e+k)'} = (e)' + (k)'$
Dminus	$-'$	$\boldsymbol{(e-k)'} = (e)' - (k)'$
Dtimes	$\cdot'$	$\boldsymbol{(e\cdot k)'} = (e)'\cdot k + e\cdot(k)'$
Dquotient	$/'$	$\boldsymbol{(e/k)'} = ((e)'\cdot k - e\cdot(k)')/k^2$
Dcompose	$\circ'$	$[y := g(x)][y' := 1]\big(\boldsymbol{(f(g(x)))'} = (f(y))'\cdot(g(x))'\big)$

12: Bellerophon tactic language operators for proof search

Bellerophon	Operation	Effect
`s ; t`	sequential composition	run `t` on the output of `s`, failing if either fail
`s | t`	alternative choice	run `t` if applying `s` failed, failing if both fail
`t*`	saturating repetition	repeat tactic `t` until nothing changes any more
`t*n`	fixed repetition	repeat tactic `t` exactly `n` times, failing if any of those repetitions fail
`<(t1,..,tn)`	branching	run tactic `ti` on branch `i`, failing if any fail or if branches $\neq$`n`
`doall(t)`	all branches	run tactic `t` on all branches `i`, failing if that fails on any branch
`t(j)`	at position	apply tactic `t` at position `j` of the sequent
`t(j,{‘e‘})`	at position	apply tactic `t` to expression `e`, which is at position `j` of the sequent
`1`	succedent position	position of first succedent formula. Similar `2`, `3`, ..., `'Rlast`
`-1`	antecedent position	position of first antecedent formula. Similar `-2`, `-3`, ..., `'Llast`
`-4.0.1`	subposition	second child of first child of fourth antecedent formula Similar `4.1`
`'R`	search succedent	first applicable succedent position (where formula `e` is, if specified)
`'L`	search antecedent	first applicable antecedent position (where formula `e` is, if specified)

```
/* Tactic "master" suffices. Mix explicit and searchy tactic for above example. */
implyR(1) ; andL('L)* ; loop({`v>=0`}, 1) ; <(      /* < splits separate branches */
  id,                                 /* initial case: prove by identity v>=0 |- v>=0 */
  QE,                                 /* postcondition: prove by real arithmetic QE */
  /* induction step: decomposes hybrid program semi-explicitly */
  composeb(1) ; solve(1.1) ; choiceb(1) ; andR(1) ; <(    /* controller branches */
    composeb(1) ; testb(1) ; master,      /* decompose some steps then ask master */
    choiceb(1) ; andR(1) ; doall(assignb(1) ; QE)  /* doall same on all branches */
  )
)
```


[1] André Platzer. *Logical Foundations of Cyber-Physical Systems*. Springer, Switzerland, 2018. URL: http://www.springer.com/978-3-319-63587-3, doi:10.1007/978-3-319-63588-0.

[2] Nathan Fulton, Stefan Mitsch, Jan-David Quesel, Marcus Völp, and André Platzer. KeYmaera X: An axiomatic tactical theorem prover for hybrid systems. In Amy Felty and Aart Middeldorp, editors, *CADE*, volume 9195 of *LNCS*, pages 527–538, Berlin, 2015. Springer. doi:10.1007/978-3-319-21401-6_36.

[3] André Platzer. A complete uniform substitution calculus for differential dynamic logic. *J. Autom. Reas.*, 59(2):219–265, 2017. doi:10.1007/s10817-016-9385-1.

[4] Nathan Fulton, Stefan Mitsch, Brandon Bohrer, and André Platzer. Bellerophon: Tactical theorem proving for hybrid systems. In Mauricio Ayala-Rincón and César A. Muñoz, editors, *ITP*, volume 10499 of *LNCS*, pages 207–224. Springer, 2017. doi:10.1007/978-3-319-66107-0_14.