The Secret for Simpler Sound Hybrid Systems Provers
Outline

1. CPS are Multi-Dynamical Systems
   - Uniform Substitution Calculus
   - Axiom vs. Axiom Schema
   - Uniform Substitutions
   - Uniform Substitution Lemmas
   - Differential Axioms
   - Differential Invariants
   - Examples

2. Uniform Substitution Calculus for Differential Dynamic Logic
   - Uniform Substitution Calculus
   - Axiom vs. Axiom Schema
   - Uniform Substitutions
   - Uniform Substitution Lemmas
   - Differential Axioms
   - Differential Invariants
   - Examples

3. Differential-form Differential Dynamic Logic
   - Syntax
   - Semantics
   - Differential Substitution Lemmas
   - Contextual Congruences
   - Parametric Computational Proofs
   - Static Semantics

4. Summary
1. CPS are Multi-Dynamical Systems

2. Uniform Substitution Calculus for Differential Dynamic Logic
   - Uniform Substitution Calculus
   - Axiom vs. Axiom Schema
   - Uniform Substitutions
   - Uniform Substitution Lemmas
   - Differential Axioms
   - Differential Invariants
   - Examples

3. Differential-form Differential Dynamic Logic
   - Syntax
   - Semantics
   - Differential Substitution Lemmas
   - Contextual Congruences
   - Parametric Computational Proofs
   - Static Semantics

4. Summary
Can you trust a computer to control physics?

Safety guarantees require analytic foundations.

Foundations revolutionized digital computer science & our society.

Need even stronger foundations when software reaches out into our physical world.

How can we provide people with cyber-physical systems they can bet their lives on? — Jeannette Wing

Cyber-physical Systems

CPS combine cyber capabilities with physical capabilities to solve problems that neither part could solve alone.
Can you trust a computer to control physics?

Rationale

1. Safety guarantees require analytic foundations.
2. Foundations revolutionized digital computer science & our society.
3. Need even stronger foundations when software reaches out into our physical world.

How can we provide people with cyber-physical systems they can bet their lives on? — Jeannette Wing

Cyber-physical Systems

CPS combine cyber capabilities with physical capabilities to solve problems that neither part could solve alone.
CPSs are Multi-Dynamical Systems

CPS Dynamics
CPS are characterized by multiple facets of dynamical systems.

CPS Compositions
CPS combine multiple simple dynamical effects.

Tame Parts
Exploiting compositionality tames CPS complexity.
Challenge (CPS)

Fixed rule describing state evolution with both

- Discrete dynamics (control decisions)
- Continuous dynamics (differential equations)
Challenge (CPS)

Fixed rule describing state evolution with both

- Discrete dynamics (control decisions)
- Continuous dynamics (differential equations)
\[ x \neq o \land b > 0 \rightarrow \left[ \begin{array}{l} \text{init} \\
\text{discrete control} \\
\text{ODE} \\
\text{post} \\
\end{array} \right] \]
Dynamic Logics for Dynamical Systems

- **differential dynamic logic**
  \[ d\mathcal{L} = DL + HP \]

- **differential game logic**
  \[ d\mathcal{G}L = GL + HG \]

- **stochastic differential DL**
  \[ Sd\mathcal{L} = DL + SHP \]

- **quantified differential DL**
  \[ Qd\mathcal{L} = FOL + DL + QHP \]

\[ [\alpha] \phi \overset{\alpha}{\longrightarrow} \phi \]

\[ \langle \alpha \rangle \phi \overset{\phi}{\longrightarrow} \phi \]

\[ JAR'08, CADE'11, LMCS'12, LICS'12 \quad \text{LICS'12, CADE'15, TOCL'15} \]
Q: How to build a prover with a small soundness-critical core?
A: Uniform substitution [Church]

Q: How to enable flexible yet sound reasoning?
A: Axioms with local meaning [Philosophy, Algebraic Geometry]

Q: What’s the local meaning of a differential equation?
A: Differential forms [Differential Geometry]

Q: How to do hybrid systems proving?
A: Uniform substitution calculus for differential dynamic logic

Q: What’s the impact of uniform substitution on a prover core?
A: 65,989 ↘ 1,677 LOC (2.5%) [KeYmaera X]
1. CPS are Multi-Dynamical Systems

2. Uniform Substitution Calculus for Differential Dynamic Logic
   - Uniform Substitution Calculus
   - Axiom vs. Axiom Schema
   - Uniform Substitutions
   - Uniform Substitution Lemmas
   - Differential Axioms
   - Differential Invariants
   - Examples

3. Differential-form Differential Dynamic Logic
   - Syntax
   - Semantics
   - Differential Substitution Lemmas
   - Contextual Congruences
   - Parametric Computational Proofs
   - Static Semantics

4. Summary
Differential Dynamic Logic: Axiomatization

\[ := \] \[ x := \theta \phi(x) \leftrightarrow \phi(\theta) \] \hspace{1cm} \text{(\(\theta\) free for \(x\) in \(\phi\))}

\[ ? \] \[ ?\chi \phi \leftrightarrow (\chi \rightarrow \phi) \]

\[ \cup \] \[ \alpha \cup \beta \phi \leftrightarrow [\alpha] \phi \land [\beta] \phi \]

\[ ; \] \[ \alpha; \beta \phi \leftrightarrow [\alpha][\beta] \phi \]

\[ * \] \[ \alpha^* \phi \leftrightarrow \phi \land [\alpha][\alpha^*] \phi \]

\[ K \] \[ \alpha(\phi \rightarrow \psi) \rightarrow ([\alpha] \phi \rightarrow [\alpha] \psi) \]

\[ I \] \[ \alpha^*(\phi \rightarrow [\alpha] \phi) \rightarrow (\phi \rightarrow [\alpha^*] \phi) \]

\[ V \] \[ \phi \rightarrow [\alpha] \phi \] \hspace{1cm} \text{\((FV(\phi) \cap BV(\alpha) = \emptyset)\)}

\[ ' \] \[ x' = \theta \phi \leftrightarrow \forall t \geq 0 [x := x(t)] \phi \] \hspace{1cm} \text{\((t\) fresh and \(x'(t) = \theta)\)}
Differential Dynamic Logic: Axioms

[\equiv] \ [x := f]p(x) \leftrightarrow p(f)

[?] \ \[?q]p \leftrightarrow (q \Rightarrow p)

[\bigcup] \ [a \cup b]p(\bar{x}) \leftrightarrow [a]p(\bar{x}) \land [b]p(\bar{x})

[;] \ [a; b]p(\bar{x}) \leftrightarrow [a][b]p(\bar{x})

[*] \ [a^*]p(\bar{x}) \leftrightarrow p(\bar{x}) \land [a][a^*]p(\bar{x})

K \ [a](p(\bar{x}) \Rightarrow q(\bar{x})) \Rightarrow ([a]p(\bar{x}) \Rightarrow [a]q(\bar{x}))

I \ [a^*](p(\bar{x}) \Rightarrow [a]p(\bar{x})) \Rightarrow (p(\bar{x}) \Rightarrow [a^*]p(\bar{x}))

\forall \ p \Rightarrow [a]p
\[
[x := f]p(x) \leftrightarrow p(f)
\]

\[
[?q]p \leftrightarrow (q \rightarrow p)
\]

\[
[a \cup b]p(\bar{x}) \leftrightarrow [a]p(\bar{x}) \land [b]p(\bar{x})
\]

\[
[a; b]p(\bar{x}) \leftrightarrow [a][b]p(\bar{x})
\]

\[
[a^*]p(\bar{x}) \leftrightarrow p(\bar{x}) \land [a][a^*]p(\bar{x})
\]

\[
[a](p(\bar{x}) \rightarrow q(\bar{x})) \rightarrow ([a]p(\bar{x}) \rightarrow [a]q(\bar{x}))
\]

\[
[a^*](p(\bar{x}) \rightarrow [a]p(\bar{x})) \rightarrow (p(\bar{x}) \rightarrow [a^*]p(\bar{x}))
\]

\[
p \rightarrow [a]p
\]

\[
[\colon] [x := \theta]p(x) \leftrightarrow \phi(\theta)
\]

\[
[?] [?\chi]p \leftrightarrow (\chi \rightarrow \phi)
\]

\[
[\cup] [\alpha \cup \beta]p \leftrightarrow [\alpha]p \land [\beta]p
\]

\[
[;] [\alpha; \beta]p \leftrightarrow [\alpha][\beta]p
\]

\[
[*] [\alpha^*]p \leftrightarrow \phi \land [\alpha][\alpha^*]p
\]

\[
[\star] [\alpha^*]p \leftrightarrow \phi \land [\alpha][\alpha^*]p
\]

\[
\Box [\alpha](\phi \rightarrow \psi) \rightarrow ([\alpha]\phi \rightarrow [\alpha]\psi)
\]

\[
[\star][\alpha^*](\phi \rightarrow [\alpha]\phi) \rightarrow (\phi \rightarrow [\alpha^*]p)
\]

\[
\forall \phi \rightarrow [\alpha]\phi
\]

\[
[\prime] [x' = \theta]p \leftrightarrow \forall t \geq 0 [x := x(t)]p
\]

CADE’15

LICS’12
Differential Dynamic Logic: Comparison

\[ x := f \] \( p(x) \leftrightarrow p(f) \)

\[ ?q \] \( p \leftrightarrow (q \rightarrow p) \)

Axiom

\[ a \cup b \] \( p(x) \leftrightarrow [a]p(x) \land [b]p(x) \)

[\( \cup \)] \[ \alpha \cup \beta \] \( \phi \leftrightarrow [\alpha] \phi \land [\beta] \phi \)

Schema

\[ a; b \] \( p(x) \leftrightarrow [a][b]p(x) \)

[\( ; \)] \[ \alpha; \beta \] \( \phi \leftrightarrow [\alpha][\beta] \phi \)

\[ a^* \] \( p(x) \leftrightarrow p(x) \land [a][a^*]p(x) \)

[\( ^* \)] \[ \alpha^* \] \( \phi \leftrightarrow \phi \land [\alpha][\alpha^*] \phi \)

\[ a \] \( (p(x) \rightarrow q(x)) \rightarrow ([a]p(x) \rightarrow [a]q(x)) \)

K \[ \alpha \] \( (\phi \rightarrow \psi) \rightarrow ([\alpha] \phi \rightarrow [\alpha] \psi) \)

Schema

\[ a^* \] \( (p(x) \rightarrow [a^*]p(x)) \rightarrow (p(x) \rightarrow [a^*]p(x)) \)

I \[ \alpha^* \] \( (\phi \rightarrow [a^*] \phi) \rightarrow (\phi \rightarrow [\alpha^*] \phi) \)

V \[ \phi \rightarrow [\alpha] \phi \]

\[ \theta \] \( [\alpha] \phi \rightarrow \forall t \geq 0 [x := x(t)] \phi \)

CADE’15

André Platzer (CMU)

FCPS / 22: Axioms & Uniform Substitutions

LICS’12

10 / 44
Axiom vs. Axiom Schema

Axiom

\[ a \cup b \, p(\bar{x}) \leftrightarrow [a]p(\bar{x}) \land [b]p(\bar{x}) \]

Schema

\[ [\alpha \cup \beta]\phi \leftrightarrow [\alpha]\phi \land [\beta]\phi \]

Axiom

\[ p \rightarrow [a]p \]

Schema

\[ \phi \rightarrow [\alpha]\phi \ldots \]
Axiom vs. Axiom Schema

\[ [a \cup b]p(\bar{x}) \leftrightarrow [a]p(\bar{x}) \land [b]p(\bar{x}) \]

\[ [\alpha \cup \beta]\phi \leftrightarrow [\alpha]\phi \land [\beta]\phi \]

- Axiom:
  \[ p \to [a]p \]
  - Pattern match formulas for shape \( \alpha \cup \beta \)

- Schema:
  \[ \phi \to [\alpha]\phi \ldots \]
  - Same instance of \( \phi \) in all places
  - Placeholder \( \alpha \) schema variable matcher

André Platzer (CMU)
FCPS / 22: Axioms & Uniform Substitutions
11 / 44
Axiom vs. Axiom Schema

Axiom

\[ [a \cup b]p(\bar{x}) \leftrightarrow [a]p(\bar{x}) \land [b]p(\bar{x}) \]

Except for special vs. degenerate instances, rule out by side conditions

p \to [a]p

Schema

\[ [\alpha \cup \beta]\phi \leftrightarrow [\alpha]\phi \land [\beta]\phi \]

Same instance of \( \phi \) in all places

\[ \phi \to [\alpha]\phi \ldots \]

\bullet \quad x = 0 \to [y' = 5]x = 0

\bullet \quad x = y \to [y' = 5]x = y

\bullet \quad x = z \to [y' = 5]x = z
Axiom vs. Axiom Schema

\[ [a \cup b]p(\bar{x}) \leftrightarrow [a]p(\bar{x}) \land [b]p(\bar{x}) \]

Axiom

\[ [\alpha \cup \beta]\phi \leftrightarrow [\alpha]\phi \land [\beta]\phi \]

Schema

Pattern match formulas for shape \( \alpha \cup \beta \)

Placeholder \( \alpha \) schema variable matcher

Same instance of \( \phi \) in all places

\[ p \rightarrow [a]p \]

\[ \phi \rightarrow [\alpha]\phi \ldots \]

special vs. degenerate instances

\[ \square x = 0 \rightarrow [y' = 5]x = 0 \]

\[ \times x = y \rightarrow [y' = 5]x = y \]

\[ \square x = z \rightarrow [y' = 5]x = z \]

rule out by side conditions
Axiom vs. Axiom Schema: Formula vs. Algorithm

- **Axiom**
  - $[a \cup b]p(\bar{x}) \leftrightarrow [a]p(\bar{x}) \land [b]p(\bar{x})$
  - Generic formula. No exceptions.
  - $p \rightarrow [a]p$
  - Axiom

- **Schema**
  - $[\alpha \cup \beta]\phi \leftrightarrow [\alpha]\phi \land [\beta]\phi$
  - Pattern match formulas for shape $\alpha \cup \beta$
  - Placeholder $\alpha$ schema variable matcher
  - Same instance of $\phi$ in all places
  - $\phi \rightarrow [\alpha]\phi \ldots$
  - Schema

- **1 Formula**

- **Algorithm**

- **Pattern**

- **Same instance of $\phi$ in all places**

- **Placeholder $\alpha$ schema variable matcher**

- **Generic formula. No exceptions.**

- **Special vs. degenerate instances**
  - $\checkmark$ $x = 0 \rightarrow [y' = 5]x = 0$
  - $\times$ $x = y \rightarrow [y' = 5]x = y$
  - $\checkmark$ $x = z \rightarrow [y' = 5]x = z$

- **Rule out by side conditions**
Generic Formulas in Axioms are like Generic Points

An analogy from algebraic geometry

Axiom schemata with side conditions are like concrete points

\[ \exists x \ ax^2 + bx + c = 0 \text{ iff } b^2 \geq 4ac \text{ except } a = 0 \]

This Way

André Platzer (CMU)

FCPS / 22: Axioms & Uniform Substitutions
Generic Formulas in Axioms are like Generic Points

An analogy from algebraic geometry

Axiom schemata with side conditions are like concrete points

\[ \exists x \ ax^2 + bx + c = 0 \text{ iff } b^2 \geq 4ac \text{ except } a = 0 \text{ except } b = 0 \]
Generic Formulas in Axioms are like Generic Points

An analogy from algebraic geometry

Axiom schemata with side conditions are like concrete points

$$\exists x \ ax^2 + bx + c = 0 \iff b^2 \geq 4ac \text{ except } a = 0 \text{ except } b = 0 \text{ except } c = 0$$

This Way
Generic Formulas in Axioms are like Generic Points

An analogy from algebraic geometry

Axiom schemata with side conditions are like concrete points

$$\exists x \ ax^2 + bx + c = 0 \iff b^2 \geq 4ac \text{ except } a = 0 \text{ except } b = 0 \text{ except } c = 0$$

This Way

Axioms

Generic formulas in axioms are like generic points

$$ax^2 + bx + c = 0 \iff x = -b \pm \sqrt{b^2 - 4ac} / (2a)$$

Paying attention during substitutions to avoid degenerates (no /0, $\sqrt{-1}$)
✓ Soundness easier: literal formula, not instantiation mechanism
✓ An axiom is one formula. Axiom schema is a decision algorithm.
✓ Generic formula, not some shape with characterization of exceptions
✓ No schema variable or meta variable algorithms
✓ No matching mechanisms / unification in prover kernel
✓ No side condition subtlety or occurrence pattern checks (per schema)
× Need other means of instantiating axioms: uniform substitution (US)
✓ US + renaming: isolate static semantics
✓ US independent from axioms: modular logic vs. prover separation
✓ More flexible by syntactic contextual equivalence
× Extra proofs branches since instantiation is explicit proof step
Axioms vs. Axiom Schemata: Philosophy Affects Provers

✓ Soundness easier: literal formula, not instantiation mechanism
✓ An axiom is one formula. Axiom schema is a decision algorithm.
✓ Generic formula, not some shape with characterization of exceptions
✓ No schema variable or meta variable algorithms
✓ No matching mechanisms / unification in prover kernel
✓ No side condition subtlety or occurrence pattern checks (per schema)

✗ Need other means of instantiating axioms: uniform substitution (US)
✓ US + renaming: isolate static semantics
✓ US independent from axioms: modular logic vs. prover separation
✓ More flexible by syntactic contextual equivalence

✗ Extra proofs branches since instantiation is explicit proof step

∑ Net win for soundness since significantly simpler prover
<table>
<thead>
<tr>
<th></th>
<th>≈LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeYmaera X</td>
<td>1677</td>
</tr>
<tr>
<td>KeYmaera</td>
<td>65 989</td>
</tr>
<tr>
<td>Key</td>
<td>51 328</td>
</tr>
<tr>
<td>HOL Light</td>
<td>396</td>
</tr>
<tr>
<td>Isabelle/Pure</td>
<td>8 113</td>
</tr>
<tr>
<td>Nuprl</td>
<td>15 000 + 50 000</td>
</tr>
<tr>
<td>Coq</td>
<td>20 000</td>
</tr>
<tr>
<td>HSolver</td>
<td>20 000</td>
</tr>
<tr>
<td>Flow*</td>
<td>25 000</td>
</tr>
<tr>
<td>PHAVer</td>
<td>30 000</td>
</tr>
<tr>
<td>dReal</td>
<td>50 000 + millions</td>
</tr>
<tr>
<td>SpaceEx</td>
<td>100 000</td>
</tr>
<tr>
<td>HyCreate2</td>
<td>6 081 + user model analysis</td>
</tr>
</tbody>
</table>

Disclaimer: These self-reported estimates of the soundness-critical lines of code + rules are to be taken with a grain of salt. Different languages, capabilities, styles...
Uniform Substitution

Theorem (Soundness)

\[ \text{US} \quad \frac{\phi}{\sigma(\phi)} \]

provided \( \text{FV}(\sigma|\Sigma(\theta)) \cap \text{BV}(\otimes(\cdot)) = \emptyset \) for each operation \( \otimes(\theta) \) in \( \phi \)

i.e. bound variables \( U = \text{BV}(\otimes(\cdot)) \) of operator \( \otimes \)
are not free in the substitution on its argument \( \theta \) \hfill (U-admissible)

\[
\text{US} \quad \frac{[a \cup b]p(\bar{x}) \leftrightarrow [a]p(\bar{x}) \land [b]p(\bar{x})}{[x := x + 1 \cup x' = 1]x \geq 0 \leftrightarrow [x := x + 1]x \geq 0 \land [x' = 1]x \geq 0}
\]
Uniform Substitution

**Theorem (Soundness)**

\[ \text{US} \quad \frac{\phi}{\sigma(\phi)} \]

provided \( FV(\sigma|\Sigma(\theta)) \cap BV(\otimes(\cdot)) = \emptyset \) for each operation \( \otimes(\theta) \) in \( \phi \)

i.e. bound variables \( U = BV(\otimes(\cdot)) \) of operator \( \otimes \)
are not free in the substitution on its argument \( \theta \) \( (U\text{-admissible}) \)
Uniform substitution \( \sigma \) replaces all occurrences of \( p(\theta) \) for any \( \theta \) by \( \psi(\theta) \)
function \( f(\theta) \) for any \( \theta \) by \( \eta(\theta) \)
quantifier \( C(\phi) \) for any \( \phi \) by \( \psi(\theta) \)
program const. \( a \) by \( \alpha \)

\[
\begin{align*}
\text{US} \quad [a \cup b]p(\bar{x}) & \iff [a]p(\bar{x}) \land [b]p(\bar{x}) \\
[x := x + 1 \cup x' = 1]x \geq 0 & \iff [x := x + 1]x \geq 0 \land [x' = 1]x \geq 0
\end{align*}
\]
Uniform Substitution: Definition expanded explicitly

\[
\begin{align*}
\sigma(f(\theta)) &= \text{def} \quad \text{for function symbol } f \in \sigma \\
\sigma(\theta + \eta) &= \\
\sigma((\theta)') &= \\
\sigma(p(\theta)) &\equiv \quad \text{for predicate symbol } p \in \sigma \\
\sigma(C(\phi)) &\equiv \\
\sigma(\phi \land \psi) &\equiv \\
\sigma(\forall x \phi) &= \\
\sigma(\lceil \alpha \rfloor \phi) &= \\
\sigma(a) &\equiv \quad \text{for program constant } a \in \sigma \\
\sigma(x := \theta) &\equiv \\
\sigma(x' = f(x) \& Q) &\equiv \\
\sigma(\alpha \cup \beta) &\equiv \\
\sigma(\alpha; \beta) &\equiv \\
\sigma(\alpha^*) &\equiv
\end{align*}
\]
Uniform Substitution: Definition expanded explicitly

\[
\begin{align*}
\sigma(f(\theta)) &= (\sigma(f))(\sigma(\theta)) \\
&\overset{\text{def}}{=} \{ \cdot \mapsto \sigma(\theta) \}(\sigma f(\cdot)) \\
\sigma(\theta + \eta) &= \\
\sigma((\theta)') &= \\
\sigma(p(\theta)) &\equiv \\
\sigma(C(\phi)) &\equiv \\
\sigma(\phi \land \psi) &\equiv \\
\sigma(\forall x \phi) &= \\
\sigma([\alpha] \phi) &= \\
\sigma(a) &\equiv \\
\sigma(x := \theta) &\equiv \\
\sigma(x' = f(x) \& Q) &\equiv \\
\sigma(\alpha \cup \beta) &\equiv \\
\sigma(\alpha; \beta) &\equiv \\
\sigma(\alpha^*) &\equiv \\
\end{align*}
\]

for function symbol \( f \in \sigma \)

for predicate symbol \( p \in \sigma \)

for program constant \( a \in \sigma \)
Uniform Substitution: Definition expanded explicitly

\[ \sigma(f(\theta)) = (\sigma(f))(\sigma(\theta)) \]
\[ \text{def} \equiv \{ \cdot \mapsto \sigma(\theta)\}(\sigma f(\cdot)) \]

for function symbol \( f \in \sigma \)

\[ \sigma(\theta + \eta) = \sigma(\theta) + \sigma(\eta) \]

\[ \sigma((\theta)') = \]

\[ \sigma(p(\theta)) \equiv \]

for predicate symbol \( p \in \sigma \)

\[ \sigma(C(\phi)) \equiv \]

\[ \sigma(\phi \land \psi) \equiv \]

\[ \sigma(\forall x \phi) = \]

\[ \sigma([\alpha]\phi) = \]

\[ \sigma(a) \equiv \]

for program constant \( a \in \sigma \)

\[ \sigma(x := \theta) \equiv \]

\[ \sigma(x' = f(x) \land Q) \equiv \]

\[ \sigma(\alpha \cup \beta) \equiv \]

\[ \sigma(\alpha; \beta) \equiv \]

\[ \sigma(\alpha^*) \equiv \]
Uniform Substitution: Definition expanded explicitly

\[ \sigma(f(\theta)) = (\sigma(f))(\sigma(\theta)) = \begin{array}{c} \text{def} \\ \equiv \{ \cdot \mapsto \sigma(\theta) \}(\sigma f(\cdot)) \end{array} \]

for function symbol \( f \in \sigma \)

\[ \sigma(\theta + \eta) = \sigma(\theta) + \sigma(\eta) \]

\[ \sigma((\theta)') = (\sigma(\theta))' \]

if \( \sigma \mathcal{V} \cup \mathcal{V}' \)-admissible for \( \theta \)

\[ \sigma(p(\theta)) \equiv \]

for predicate symbol \( p \in \sigma \)

\[ \sigma(C(\phi)) \equiv \]

\[ \sigma(\phi \land \psi) \equiv \]

\[ \sigma(\forall x \phi) = \]

\[ \sigma([\alpha]\phi) = \]

\[ \sigma(a) \equiv \]

for program constant \( a \in \sigma \)

\[ \sigma(x := \theta) \equiv \]

\[ \sigma(x' = f(x) \& Q) \equiv \]

\[ \sigma(\alpha \cup \beta) \equiv \]

\[ \sigma(\alpha; \beta) \equiv \]

\[ \sigma(\alpha^*) \equiv \]
<table>
<thead>
<tr>
<th>Expression</th>
<th>Definition</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma(f(\theta))$</td>
<td>$(\sigma(f))(\sigma(\theta))$</td>
<td>for function symbol $f \in \sigma$</td>
</tr>
<tr>
<td>$\sigma(\theta + \eta)$</td>
<td>$\sigma(\theta) + \sigma(\eta)$</td>
<td></td>
</tr>
<tr>
<td>$\sigma((\theta)')$</td>
<td>$(\sigma(\theta))'$</td>
<td>if $\sigma \mathcal{V} \cup \mathcal{V}'$-admissible for $\theta$</td>
</tr>
<tr>
<td>$\sigma(p(\theta))$</td>
<td>$(\sigma(p))(\sigma(\theta))$</td>
<td>for predicate symbol $p \in \sigma$</td>
</tr>
<tr>
<td>$\sigma(C(\phi))$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\sigma(\phi \land \psi)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\sigma(\forall x \phi)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\sigma([\alpha]\phi)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\sigma(a)$</td>
<td></td>
<td>for program constant $a \in \sigma$</td>
</tr>
<tr>
<td>$\sigma(x := \theta)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\sigma(x' = f(x) \land Q)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\sigma(\alpha \cup \beta)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\sigma(\alpha; \beta)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\sigma(\alpha^*)$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Uniform Substitution: Definition expanded explicitly

\[ \sigma(f(\theta)) = (\sigma(f))(\sigma(\theta)) \]
\[ \text{def} \]
\[ \equiv \{ \cdot \mapsto \sigma(\theta)\}(\sigma f(\cdot)) \]

for function symbol \( f \in \sigma \)

\[ \sigma(\theta + \eta) = \sigma(\theta) + \sigma(\eta) \]

if \( \sigma \mathcal{V} \cup \mathcal{V}' \)-admissible for \( \theta \)

\[ \sigma((\theta)') = (\sigma(\theta))' \]

\[ \sigma(p(\theta)) \equiv (\sigma(p))(\sigma(\theta)) \]

for predicate symbol \( p \in \sigma \)

\[ \sigma(C(\phi)) \equiv \sigma(C)(\sigma(\phi)) \]

if \( \sigma \mathcal{V} \cup \mathcal{V}' \)-admissible for \( \phi, C \in \sigma \)

\[ \sigma(\phi \land \psi) \equiv \sigma(\phi \land \psi) \]

\[ \sigma(\forall x \phi) = \sigma(\forall x \phi) \]

\[ \sigma([\alpha] \phi) = \sigma([\alpha] \phi) \]

\[ \sigma(a) \equiv \sigma(a) \]

for program constant \( a \in \sigma \)

\[ \sigma(x := \theta) \equiv \sigma(x := \theta) \]

\[ \sigma(x' = f(x) \land Q) \equiv \sigma(x' = f(x) \land Q) \]

\[ \sigma(\alpha \cup \beta) \equiv \sigma(\alpha \cup \beta) \]

\[ \sigma(\alpha; \beta) \equiv \sigma(\alpha; \beta) \]

\[ \sigma(\alpha^*) \equiv \sigma(\alpha^*) \]
Uniform Substitution: Definition expanded explicitly

\[
\begin{align*}
\sigma(f(\theta)) &= (\sigma(f))(\sigma(\theta)) \\
& \overset{\text{def}}{=} \{ \cdot \mapsto \sigma(\theta) \}(\sigma f(\cdot)) \\
\sigma(\theta + \eta) &= \sigma(\theta) + \sigma(\eta) \\
\sigma((\theta)') &= (\sigma(\theta))' \\
\sigma(p(\theta)) &\equiv (\sigma(p))(\sigma(\theta)) \\
\sigma(C(\phi)) &\equiv \sigma(C)(\sigma(\phi)) \\
\sigma(\phi \land \psi) &\equiv \sigma(\phi) \land \sigma(\psi) \\
\sigma(\forall x \phi) &\equiv \\
\sigma([\alpha] \phi) &\equiv
\end{align*}
\]

for function symbol \( f \in \sigma \)

if \( \sigma \mathcal{V} \cup \mathcal{V}'\)-admissible for \( \theta \)

for predicate symbol \( p \in \sigma \)

if \( \sigma \mathcal{V} \cup \mathcal{V}'\)-admissible for \( \phi, C \in \sigma \)

for program constant \( a \in \sigma \)

\[
\begin{align*}
\sigma(a) &\equiv \\
\sigma(x := \theta) &\equiv \\
\sigma(x' = f(x) \& Q) &\equiv \\
\sigma(\alpha \cup \beta) &\equiv \\
\sigma(\alpha; \beta) &\equiv \\
\sigma(\alpha^*) &\equiv
\end{align*}
\]
\[ \sigma(f(\theta)) = (\sigma(f))(\sigma(\theta)) \]
for function symbol \( f \in \sigma \)

\[
\begin{align*}
\sigma(\theta + \eta) &= \sigma(\theta) + \sigma(\eta) \\
\sigma((\theta)') &= (\sigma(\theta))' 
\end{align*}
\]
for predicate symbol \( p \in \sigma \)

\[
\begin{align*}
\sigma(C(\phi)) &= \sigma(C)(\sigma(\phi)) \\
\sigma(\phi \land \psi) &= \sigma(\phi) \land \sigma(\psi) \\
\sigma(\forall x \phi) &= \forall x \sigma(\phi) \\
\sigma([\alpha]\phi) &= 
\end{align*}
\]
if \( \sigma \bigcup \mathcal{V}' \)-admissible for \( \phi \)

\[
\sigma(a) \equiv 
\]
for program constant \( a \in \sigma \)

\[
\begin{align*}
\sigma(x := \theta) &
\end{align*}
\]

\[
\begin{align*}
\sigma(x' = f(x) \& Q) &
\end{align*}
\]

\[
\begin{align*}
\sigma(\alpha \cup \beta) &
\end{align*}
\]

\[
\begin{align*}
\sigma(\alpha; \beta) &
\end{align*}
\]

\[
\begin{align*}
\sigma(\alpha^*) &
\end{align*}
\]
Uniform Substitution: Definition expanded explicitly

\[ \sigma(f(\theta)) = (\sigma(f))(\sigma(\theta)) \]
\[ \overset{\text{def}}{=} \{ \cdot \mapsto \sigma(\theta) \}(\sigma f(\cdot)) \]

for function symbol \( f \in \sigma \)

\[ \sigma(\theta + \eta) = \sigma(\theta) + \sigma(\eta) \]

if \( \sigma \mathcal{V} \cup \mathcal{V}' \text{-admissible for } \theta \)

\[ \sigma((\theta)') = (\sigma(\theta))' \]

\[ \sigma(p(\theta)) \equiv (\sigma(p))(\sigma(\theta)) \]

for predicate symbol \( p \in \sigma \)

\[ \sigma(C(\phi)) \equiv \sigma(C)(\sigma(\phi)) \]

if \( \sigma \mathcal{V} \cup \mathcal{V}' \text{-admissible for } \phi, \ C \in \sigma \)

\[ \sigma(\phi \land \psi) \equiv \sigma(\phi) \land \sigma(\psi) \]

\[ \sigma(\forall x \phi) = \forall x \sigma(\phi) \]

if \( \sigma \{ x \} \text{-admissible for } \phi \)

\[ \sigma([\alpha] \phi) = [\sigma(\alpha)]\sigma(\phi) \]

if \( \sigma \text{ BV}(\sigma(\alpha)) \text{-admissible for } \phi \)

\[ \sigma(a) \equiv \]

\[ \sigma(x := \theta) \equiv \]

\[ \sigma(x' = f(x) \& Q) \equiv \]

\[ \sigma(\alpha \cup \beta) \equiv \]

\[ \sigma(\alpha; \beta) \equiv \]

\[ \sigma(\alpha^*) \equiv \]

for program constant \( a \in \sigma \)
Uniform Substitution: Definition expanded explicitly

\[
\sigma(f(\theta)) = (\sigma(f))(\sigma(\theta))
\]
for function symbol \(f \in \sigma\)

\[
\sigma(\theta + \eta) = \sigma(\theta) + \sigma(\eta)
\]
if \(\sigma \mathcal{V} \cup \mathcal{V}'\)-admissible for \(\theta\)

\[
\sigma((\theta)') = (\sigma(\theta))'
\]
if \(\sigma \mathcal{V} \cup \mathcal{V}'\)-admissible for \(\theta\)

\[
\sigma(p(\theta)) \equiv (\sigma(p))(\sigma(\theta))
\]
for predicate symbol \(p \in \sigma\)

\[
\sigma(C(\phi)) \equiv \sigma(C)(\sigma(\phi))
\]
if \(\sigma \mathcal{V} \cup \mathcal{V}'\)-admissible for \(\phi, C \in \mathcal{C}\)

\[
\sigma(\phi \land \psi) \equiv \sigma(\phi) \land \sigma(\psi)
\]
if \(\sigma \mathcal{V} \cup \mathcal{V}'\)-admissible for \(\phi\)

\[
\sigma(\forall x \phi) = \forall x \sigma(\phi)
\]
if \(\sigma \{x\}\)-admissible for \(\phi\)

\[
\sigma([\alpha]\phi) = [\sigma(\alpha)]\sigma(\phi)
\]
if \(\sigma \mathcal{BV}(\sigma(\alpha))\)-admissible for \(\phi\)

\[
\sigma(a) \equiv \sigma a
\]
for program constant \(a \in \sigma\)
\[
\sigma(f(\theta)) = (\sigma(f))(\sigma(\theta)) \\
\sigma(\theta + \eta) = \sigma(\theta) + \sigma(\eta) \\
\sigma((\theta)') = (\sigma(\theta))' \\
\sigma(p(\theta)) \equiv (\sigma(p))(\sigma(\theta)) \\
\sigma(C(\phi)) \equiv \sigma(C)(\sigma(\phi)) \\
\sigma(\phi \land \psi) \equiv \sigma(\phi) \land \sigma(\psi) \\
\sigma(\forall x \phi) = \forall x \sigma(\phi) \\
\sigma([\alpha]\phi) = [\sigma(\alpha)]\sigma(\phi) \\
\sigma(a) \equiv \sigma a \\
\sigma(x := \theta) \equiv x := \sigma(\theta) \\
\sigma(x' = f(x) \& Q) \equiv \\
\sigma(\alpha \cup \beta) \equiv \\
\sigma(\alpha; \beta) \equiv \\
\sigma(\alpha^*) \equiv 
\]

for function symbol \( f \in \sigma \)

if \( \sigma \mathcal{V} \cup \mathcal{V}' \)-admissible for \( \theta \)

for predicate symbol \( p \in \sigma \)

if \( \sigma \mathcal{V} \cup \mathcal{V}' \)-admissible for \( \phi, \ C \in \sigma \)

if \( \sigma \{x\}\)-admissible for \( \phi \)

if \( \sigma \text{BV}(\sigma(\alpha))\)-admissible for \( \phi \)

for program constant \( a \in \sigma \)
Uniform Substitution: Definition expanded explicitly

\[ \sigma(f(\theta)) = (\sigma(f))(\sigma(\theta)) \]
\[ \overset{\text{def}}{=} \{ \cdot \mapsto \sigma(\theta) \}(\sigma f(\cdot)) \]
\[ \sigma(\theta + \eta) = \sigma(\theta) + \sigma(\eta) \]
\[ \sigma((\theta)') = (\sigma(\theta))' \]

if \( \sigma \mathcal{V} \cup \mathcal{V}' \)-admissible for \( \theta \)

\[ \sigma(p(\theta)) \equiv (\sigma(p))(\sigma(\theta)) \]
\[ \sigma(C(\phi)) \equiv \sigma(C)(\sigma(\phi)) \]
\[ \sigma(\phi \land \psi) \equiv \sigma(\phi) \land \sigma(\psi) \]
\[ \sigma(\forall x \phi) = \forall x \sigma(\phi) \]
\[ \sigma([\alpha]\phi) = [\sigma(\alpha)]\sigma(\phi) \]

if \( \sigma \{ x \}\)-admissible for \( \phi \)

\[ \sigma(\alpha \cup \beta) \equiv \]
\[ \sigma(\alpha; \beta) \equiv \]
\[ \sigma(\alpha^*) \equiv \]

for function symbol \( f \in \sigma \)

for predicate symbol \( p \in \sigma \)

if \( \sigma \mathcal{V} \cup \mathcal{V}' \)-admissible for \( \phi, C \in \sigma \)

if \( \sigma \mathcal{V} \cup \mathcal{V}' \)-admissible for \( \phi \)

for program constant \( a \in \sigma \)

if \( \sigma \{ x, x' \}\)-admissible for \( f(x), Q \)
Uniform Substitution: Definition expanded explicitly

\[
\begin{align*}
\sigma(f(\theta)) &= (\sigma(f))(\sigma(\theta)) \\
& \overset{\text{def}}{=} \{ \cdot \mapsto \sigma(\theta) \}(\sigma f(\cdot)) \\
\sigma(\theta + \eta) &= \sigma(\theta) + \sigma(\eta) \\
\sigma((\theta)') &= (\sigma(\theta))' \\
\sigma(p(\theta)) &\equiv (\sigma(p))(\sigma(\theta)) \\
\sigma(C(\phi)) &\equiv \sigma(C)(\sigma(\phi)) \\
\sigma(\phi \land \psi) &\equiv \sigma(\phi) \land \sigma(\psi) \\
\sigma(\forall x \phi) &\equiv \forall x \sigma(\phi) \\
\sigma(\lceil \alpha \rceil \phi) &\equiv [\sigma(\alpha)]\sigma(\phi) \\
\sigma(a) &\equiv \sigma a \\
\sigma(x := \theta) &\equiv x := \sigma(\theta) \\
\sigma(x' = f(x) \& Q) &\equiv x' = \sigma(f(x)) \& \sigma(Q) \\
\sigma(\alpha \cup \beta) &\equiv \sigma(\alpha) \cup \sigma(\beta) \\
\sigma(\alpha; \beta) &\equiv \\
\sigma(\alpha^*) &\equiv \\
\end{align*}
\]

for function symbol \( f \in \sigma \)

if \( \sigma \mathcal{V} \cup \mathcal{V}' \)-admissible for \( \theta \)

for predicate symbol \( p \in \sigma \)

if \( \sigma \mathcal{V} \cup \mathcal{V}' \)-admissible for \( \phi, C \in \mathcal{V} \)

if \( \sigma \{ x \} \)-admissible for \( \phi \)

if \( \sigma \text{ BV}(\sigma(\alpha)) \)-admissible for \( \phi \)

for program constant \( a \in \sigma \)

if \( \sigma \{ x, x' \} \)-admissible for \( f(x), Q \)
<table>
<thead>
<tr>
<th>Expression</th>
<th>Definition</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \sigma(f(\theta)) = (\sigma(f))(\sigma(\theta)) )</td>
<td>( \sigma(\theta) )</td>
<td>for function symbol ( f \in \sigma )</td>
</tr>
<tr>
<td>( \sigma(\theta + \eta) = \sigma(\theta) + \sigma(\eta) )</td>
<td>( \sigma(\theta) )</td>
<td>if ( \sigma \cup \nu )-admissible for ( \theta )</td>
</tr>
<tr>
<td>( \sigma((\theta)') = (\sigma(\theta))' )</td>
<td>( \sigma(\theta) )</td>
<td></td>
</tr>
<tr>
<td>( \sigma(p(\theta)) \equiv (\sigma(p))(\sigma(\theta)) )</td>
<td>( \sigma(\theta) )</td>
<td>for predicate symbol ( p \in \sigma )</td>
</tr>
<tr>
<td>( \sigma(C(\phi)) \equiv \sigma(C)(\sigma(\phi)) )</td>
<td>( \sigma(\theta) )</td>
<td>if ( \sigma \cup \nu )-admissible for ( \phi ), ( C \in \sigma )</td>
</tr>
<tr>
<td>( \sigma(\phi \land \psi) \equiv \sigma(\phi) \land \sigma(\psi) )</td>
<td>( \sigma(\theta) )</td>
<td></td>
</tr>
<tr>
<td>( \sigma(\forall x \phi) \equiv \forall x \sigma(\phi) )</td>
<td>( \sigma(\theta) )</td>
<td>if ( \sigma {x}-admissible for ( \phi )</td>
</tr>
<tr>
<td>( \sigma([\alpha]\phi) \equiv [\sigma(\alpha)]\sigma(\phi) )</td>
<td>( \sigma(\theta) )</td>
<td>if ( \sigma ) ( \BV(\sigma(\alpha)) )-admissible for ( \phi )</td>
</tr>
<tr>
<td>( \sigma(a) \equiv \sigma a )</td>
<td>( \sigma(\theta) )</td>
<td>for program constant ( a \in \sigma )</td>
</tr>
<tr>
<td>( \sigma(x := \theta) \equiv x := \sigma(\theta) )</td>
<td>( \sigma(\theta) )</td>
<td></td>
</tr>
<tr>
<td>( \sigma(x' = f(x) \land Q) \equiv x' = \sigma(f(\sigma(x))) \land \sigma(Q) )</td>
<td>( \sigma(\theta) )</td>
<td>if ( \sigma ) {x, x'}-admissible for ( f(x), Q )</td>
</tr>
<tr>
<td>( \sigma(\alpha \cup \beta) \equiv \sigma(\alpha) \cup \sigma(\beta) )</td>
<td>( \sigma(\theta) )</td>
<td>if ( \sigma ) ( \BV(\sigma(\alpha)) )-admissible for ( \beta )</td>
</tr>
<tr>
<td>( \sigma(\alpha^*) \equiv \sigma(\theta) )</td>
<td>( \sigma(\theta) )</td>
<td></td>
</tr>
</tbody>
</table>
\[ \sigma(f(\theta)) = (\sigma(f))(\sigma(\theta)) \quad \text{for function symbol } f \in \sigma \]
\[ \sigma(\theta + \eta) = \sigma(\theta) + \sigma(\eta) \quad \text{if } \sigma \mathcal{V} \cup \mathcal{V}' \text{-admissible for } \theta \]
\[ \sigma((\theta)') = (\sigma(\theta))' \quad \text{if } \sigma \mathcal{V} \cup \mathcal{V}' \text{-admissible for } \theta \]
\[ \sigma(p(\theta)) \equiv (\sigma(p))(\sigma(\theta)) \quad \text{for predicate symbol } p \in \sigma \]
\[ \sigma(C(\phi)) \equiv \sigma(C)(\sigma(\phi)) \quad \text{if } \sigma \mathcal{V} \cup \mathcal{V}' \text{-admissible for } \phi, C \in \sigma \]
\[ \sigma(\phi \land \psi) \equiv \sigma(\phi) \land \sigma(\psi) \]
\[ \sigma(\forall x \phi) = \forall x \sigma(\phi) \quad \text{if } \sigma \{x\} \text{-admissible for } \phi \]
\[ \sigma([\alpha]\phi) = [\sigma(\alpha)]\sigma(\phi) \quad \text{if } \sigma \mathcal{BV}(\sigma(\alpha)) \text{-admissible for } \phi \]
\[ \sigma(a) \equiv \sigma a \quad \text{for program constant } a \in \sigma \]
\[ \sigma(x := \theta) \equiv x := \sigma(\theta) \]
\[ \sigma(x' = f(x) & Q) \equiv x' = \sigma(f(x)) & \sigma(Q) \quad \text{if } \sigma \{x, x'\} \text{-admissible for } f(x), Q \]
\[ \sigma(\alpha \cup \beta) \equiv \sigma(\alpha) \cup \sigma(\beta) \quad \text{if } \sigma \mathcal{BV}(\sigma(\alpha)) \text{-admissible for } \beta \]
\[ \sigma(\alpha; \beta) \equiv \sigma(\alpha); \sigma(\beta) \quad \text{if } \sigma \mathcal{BV}(\sigma(\alpha)) \text{-admissible for } \alpha \]
\[ \sigma(\alpha^*) \equiv (\sigma(\alpha))^* \]
Uniform Substitution: Examples

\[ [x := f]p(x) \leftrightarrow p(f) \]
\[
[x := x + 1]x \neq x \leftrightarrow x + 1 \neq x
\]
\[
\sigma = \{ f \mapsto x + 1, p(\cdot) \mapsto (\cdot \neq x) \}
\]

\[ [x := f]p(x) \leftrightarrow p(f) \]
\[
[x := x^2][(z := x + z)^*; z := x + yz]y \geq x \leftrightarrow [(z := x^2 + z^*); z := x^2 + yz]y \geq x^2
\]
with \( \sigma = \{ f \mapsto x^2, p(\cdot) \mapsto [(z := \cdot + z)^*; z := \cdot + yz]y \geq \cdot \} \)

\[ p \to [a]p \]
\[
x \geq 0 \to [x' = -1]x \geq 0
\]
\[
\sigma = \{ a \mapsto x' = -1, p \mapsto x \geq 0 \}
\]

\[ (-x)^2 \geq 0 \]
\[
[x' = -1](-x)^2 \geq 0
\]
by \[ p(\bar{x}) \]
\[
[a]p(\bar{x})
\]
\[
\sigma = \{ a \mapsto x' = -1, p(\cdot) \mapsto (-\cdot)^2 \geq 0 \} \]
Uniform Substitution: Examples

\[
[x := f] p(x) \leftrightarrow p(f)
\]
\[
[x := x + 1] x \neq x \leftrightarrow x + 1 \neq x
\]
\[
\sigma = \{ f \mapsto x + 1, p(\cdot) \mapsto (\cdot \neq x) \}
\]

\[
[x := f] p(x) \leftrightarrow p(f)
\]
\[
[x := x^2][(z := x + z)^*; z := x + yz] y \geq x \leftrightarrow [(z := x^2 + z^*); z := x^2 + yz] y \geq x^2
\]
\[
\text{with } \sigma = \{ f \mapsto x^2, p(\cdot) \mapsto [(z := \cdot + z)^*; z := \cdot + yz] y \geq \cdot \}
\]

\[
p \rightarrow [a] p
\]
\[
x \geq 0 \rightarrow [x' = -1] x \geq 0
\]
\[
\sigma = \{ a \mapsto x' = -1, p \mapsto x \geq 0 \}
\]

\[
(-x)^2 \geq 0
\]
\[
[x' = -1] (-x)^2 \geq 0
\]
\[
\text{by } \frac{p(\bar{x})}{[a] p(\bar{x})}
\]
\[
\sigma = \{ a \mapsto x' = -1, p(\cdot) \mapsto (-\cdot)^2 \geq 0 \}
\]
Uniform Substitution: Examples

\[
\begin{align*}
[x := f]p(x) & \leftrightarrow p(f) \\
[x := x + 1]x & \neq x \iff x + 1 \neq x \\
\sigma & = \{ f \mapsto x + 1, p(\cdot) \mapsto (\cdot \neq x) \}
\end{align*}
\]

\[
\begin{align*}
[x := f]p(x) & \leftrightarrow p(f) \\
[x := x^2][(z := x + z)^*; z := x + yz]y \geq x & \iff [(z := x^2 + z^*); z := x^2 + yz]y \geq x^2 \\
\text{with } \sigma & = \{ f \mapsto x^2, p(\cdot) \mapsto [(z := \cdot + z)^*; z := \cdot + yz]y \geq \cdot \}
\end{align*}
\]

\[
\begin{align*}
p & \rightarrow [a]p \\
x \geq 0 & \rightarrow [x' = -1]x \geq 0 \\
\sigma & = \{ a \mapsto x' = -1, p \mapsto x \geq 0 \}
\end{align*}
\]

\[
\begin{align*}
(-x)^2 & \geq 0 \\
[x' = -1](-x)^2 & \geq 0 \\
\text{by } \frac{p(\bar{x})}{[a]p(\bar{x})} & \sigma = \{ a \mapsto x' = -1, p(\cdot) \mapsto (\cdot)^2 \geq 0 \}
\end{align*}
\]
Uniform Substitution: Examples

\[ [x := f]p(x) \leftrightarrow p(f) \]
\[ [x := x + 1]x \neq x \leftrightarrow x + 1 \neq x \]
\[ \sigma = \{ f \mapsto x + 1, p(\cdot) \mapsto (\cdot \neq x) \} \]

**Clash**

\[ [x := f]p(x) \leftrightarrow p(f) \]
\[ [x := x^2][(z := x + z)^*; z := x + yz)y \geq x \leftrightarrow [(z := x^2 + z^*); z := x^2 + yz)y \geq x^2 \]
with \( \sigma = \{ f \mapsto x^2, p(\cdot) \mapsto [(z := \cdot + z)^*; z := \cdot + yz)y \geq \cdot \} \)

\[ p \rightarrow [a]p \]
\[ x \geq 0 \rightarrow [x' = -1]x \geq 0 \]
\[ \sigma = \{ a \mapsto x' = -1, p \mapsto x \geq 0 \} \]

\[ (-x)^2 \geq 0 \]
\[ [x' = -1](-x)^2 \geq 0 \]
by
\[ \frac{p(\bar{x})}{[a]p(\bar{x})} \]
\[ \sigma = \{ a \mapsto x' = -1, p(\cdot) \mapsto (\cdot)^2 \geq 0 \} \]
### Uniform Substitution: Examples

\[
\begin{align*}
[x := f] p(x) & \leftrightarrow p(f) \\
\\text{Clash} \\
[x := x + 1] & x \neq x \leftrightarrow x + 1 \neq x
\end{align*}
\]

\[\sigma = \{ f \mapsto x + 1, p(\cdot) \mapsto (\cdot \neq x) \}\]

\[
\begin{align*}
[x := f] p(x) & \leftrightarrow p(f) \\
\text{Correct} \\
[x := x^2][(z := x+z)^*; z := x+yz] y \geq x & \leftrightarrow [(z := x^2+z^*); z := x^2+yz] y \geq x^2 \\
\text{with } \sigma = \{ f \mapsto x^2, p(\cdot) \mapsto [(z := \cdot + z)^*; z := \cdot + yz] y \geq \cdot \}\end{align*}
\]

\[
\begin{align*}
p & \rightarrow [a] p \\
x \geq 0 & \rightarrow [x' = -1] x \geq 0 \\
\sigma = \{ a \mapsto x' = -1, p \mapsto x \geq 0 \}\end{align*}
\]

\[
\begin{align*}
(-x)^2 & \geq 0 \\
[x' = -1](-x)^2 & \geq 0 \\
\text{by } \frac{p(\bar{x})}{[a] p(\bar{x})} \\
\sigma = \{ a \mapsto x' = -1, p(\cdot) \mapsto (-\cdot)^2 \geq 0 \}\end{align*}
\]
Uniform Substitution: Examples

\[
[x := f]p(x) \leftrightarrow p(f)
\]
Clash

\[
[x := x + 1]x \neq x \leftrightarrow x + 1 \neq x
\]
\[
\sigma = \{ f \mapsto x + 1, p(\cdot) \mapsto (\cdot \neq x) \}
\]

\[
[x := f]p(x) \leftrightarrow p(f)
\]
Correct

\[
[x := x^2][(z := x + z)^*; z := x + yz]y \geq x \leftrightarrow [(z := x^2 + z^*); z := x^2 + yz]y \geq x^2
\]
with \[
\sigma = \{ f \mapsto x^2, p(\cdot) \mapsto [(z := \cdot + z)^*; z := \cdot + yz]y \geq \cdot \}
\]

\[
x \geq 0 \rightarrow [x' = -1]x \geq 0
\]
BV \[a]p

Clash
\[
x \geq 0 \rightarrow [x' = -1]x \geq 0
\]

\[
(-x)^2 \geq 0
\]
by
\[
\frac{p(\bar{x})}{[a]p(\bar{x})}
\]
FV

\[
\sigma = \{ a \mapsto x' = -1, p(\cdot) \mapsto x \geq 0 \}
\]

\[
\frac{(-x)^2 \geq 0}{[x' = -1](-x)^2 \geq 0}
\]

\[
\sigma = \{ a \mapsto x' = -1, p(\cdot) \mapsto (-\cdot)^2 \geq 0 \}
\]
Uniform Substitution: Examples

\[
[x := f]p(x) \leftrightarrow p(f) \quad \text{Clash}
\]

\[
[x := x + 1]x \neq x \leftrightarrow x + 1 \neq x \\
\sigma = \{ f \mapsto x + 1, p(\cdot) \mapsto (\cdot \neq x) \}
\]

\[
[x := f]p(x) \leftrightarrow p(f) \quad \text{Correct}
\]

\[
[x := x^2][(z := x + z)^*; z := x + yz]y \geq x \leftrightarrow [(z := x^2 + z^*); z := x^2 + yz]y \geq x^2
\]

with \( \sigma = \{ f \mapsto x^2, p(\cdot) \mapsto [(z := \cdot + z)^*; z := \cdot + yz]y \geq \cdot \} \)

\[
p \to [a]p \\ x \geq 0 \to [x' = -1]x \geq 0 \\
\sigma = \{ a \mapsto x' = -1, p \mapsto x \geq 0 \}
\]

\[
(-x)^2 \geq 0 \quad \text{Correct}
\]

\[
[x' = -1](-x)^2 \geq 0 \\
\sigma = \{ a \mapsto x' = -1, p(\cdot) \mapsto (-\cdot)^2 \geq 0 \}
\]

by

\[
\frac{p(\bar{x})}{[a]p(\bar{x})}
\]
Uniform Substitution: Contextual Congruence Example

Theorem (Soundness): $\text{FV}(\sigma) = \emptyset$ implies

$$\phi_1 \ldots \phi_n$$

locally sound implies

$$\sigma(\phi_1) \ldots \sigma(\phi_n)$$

$$\sigma(\psi)$$

locally sound

The conclusion is valid in any interpretation $I$ in which the premises are.

### Example

$$p(\bar{x}) \leftrightarrow q(\bar{x})$$

$$C(p(\bar{x})) \leftrightarrow C(q(\bar{x}))$$

$$[x := x^2]x \leq 1 \leftrightarrow x^2 \leq 1$$

$$[x' = x^3 \cup x' = -1][x := x^2]x \leq 1 \leftrightarrow [x' = x^3 \cup x' = -1]x^2 \leq 1$$
### Uniform Substitution: Contextual Congruence Example

\[
\begin{align*}
CE & \quad p(\bar{x}) \leftrightarrow q(\bar{x}) \\
& \quad C(p(\bar{x})) \leftrightarrow C(q(\bar{x}))
\end{align*}
\]

\[
\begin{align*}
CE & \quad [x := x^2]x \leq 1 \leftrightarrow x^2 \leq 1 \\
& \quad [x' = x^3 \cup x' = -1][x := x^2]x \leq 1 \leftrightarrow [x' = x^3 \cup x' = -1]x^2 \leq 1
\end{align*}
\]

---

**Theorem (Soundness)**

\[
\begin{align*}
\phi_1 & \quad \ldots \quad \phi_n \\
\psi & \quad \text{locally sound} \quad \text{implies} \quad \sigma(\phi_1) & \quad \ldots & \quad \sigma(\phi_n) & \quad \sigma(\psi) \quad \text{locally sound}
\end{align*}
\]

(FV(\sigma) = \emptyset)
Uniform Substitution: Contextual Congruence Example

CE \(\frac{p(\bar{x}) \leftrightarrow q(\bar{x})}{C(p(\bar{x})) \leftrightarrow C(q(\bar{x}))}\)

\[
[x := x^2]x \leq 1 \leftrightarrow x^2 \leq 1
\]

CE \(\frac{[x' = x^3 \cup x' = -1][x := x^2]x \leq 1 \leftrightarrow [x' = x^3 \cup x' = -1]x^2 \leq 1}{[x' = x^3 \cup x' = -1][x := x^2]x \leq 1 \leftrightarrow [x' = x^3 \cup x' = -1]x^2 \leq 1}\)

Theorem (Soundness) \((FV(\sigma) = \emptyset)\)

\[
\frac{\phi_1 \ldots \phi_n \quad \psi}{\sigma(\phi_1) \ldots \sigma(\phi_n) \quad \sigma(\psi)}\text{ locally sound implies locally sound}
\]

Locally sound

The conclusion is valid in any interpretation \(I\) in which the premises are.
“Syntactic uniform substitution = semantic replacement”

**Lemma (Uniform substitution lemma)**

Uniform substitution $\sigma$ and its adjoint interpretation $\sigma^*_u l$ to $\sigma$ for $l, u$ have the same semantics:

$$[\sigma(\theta)]lu = [\theta]\sigma^*_u lu$$

$u \in [\sigma(\phi)]l$ iff $u \in [\phi]\sigma^*_u l$

$(u, w) \in [\sigma(\alpha)]l$ iff $(u, w) \in [\alpha]\sigma^*_u l$
Solving Differential Equations? By Axiom Schema?

\[ [x' = \theta] \phi \iff \forall t \geq 0 [x := x(t)] \phi \]  

\( t \) fresh and \( x'(t) = \theta \)
Solving Differential Equations? By Axiom Schema?

\[
[\forall t \geq 0 \left[ x(t) \right]] \phi \leftrightarrow \forall t \geq 0 \left[ x := x(t) \right] \phi
\]

(t fresh and \( x'(t) = \theta \))

Axiom schema with side conditions:

1. Occurs check: \( t \) fresh
2. Solution check: \( x(\cdot) \) solves the ODE \( x'(t) = \theta \) with \( x(\cdot) \) plugged in for \( x \) in \( \theta \)
3. Initial value check: \( x(\cdot) \) solves the symbolic IVP \( x(0) = x \)

Quite nontrivial soundness-critical algorithms ...
Solvi ng Differential Equations? By Axiom Schema?

\[ [\forall t \geq 0 [x(t) = x(t)]\phi \iff \forall t \geq 0 [x := x(t)]\phi \] (\text{t fresh and } x'(t) = \theta)

Axiom schema with side conditions:

1. Occurs check: \( t \) fresh
2. Solution check: \( x(\cdot) \) solves the ODE \( x'(t) = \theta \)
   with \( x(\cdot) \) plugged in for \( x \) in \( \theta \)
3. Initial value check: \( x(\cdot) \) solves the symbolic IVP \( x(0) = x \)
4. \( x(\cdot) \) covers all solutions parametrically

Quite nontrivial soundness-critical algorithms . . .
Differential Equation Axioms & Differential Axioms

DW \[ x' = f(x) & q(x)]q(x) \]

DC \[left\{ [x' = f(x) & q(x)]p(x) \leftrightarrow [x' = f(x) & q(x) \land r(x)]p(x) \right\} \leftrightarrow [x' = f(x) & q(x)]r(x) \]

DE \[ x' = f(x) & q(x)]p(x, x') \leftrightarrow [x' = f(x) & q(x)][x' := f(x)]p(x, x') \]

DI \[ x' = f(x) & q(x)]p(x) \leftarrow (q(x) \rightarrow p(x) \land [x' = f(x) & q(x)]'(p(x))' \]

DG \[ x' = f(x) & q(x)]p(x) \leftrightarrow \exists y [x' = f(x), y' = a(x)y + b(x) & q(x)]p(x) \]

DS \[ x' = f & q(x)]p(x) \leftrightarrow \forall t \geq 0 ((\forall 0 \leq s \leq t q(x + fs)) \rightarrow [x := x + ft]p(x)) \]

\[ x' := f \]p(x') \leftrightarrow p(f) \]

\[ f(x) + g(x)]' = (f(x))' + (g(x))' \]

\[ f(x) \cdot g(x)]' = (f(x))' \cdot g(x) + f(x) \cdot (g(x))' \]

\[ y := g(x)][y' := 1)((f(g(x)))' = (f(y))' \cdot (g(x))') \]
Differential Invariants for Differential Equations

Differential Invariant

\[ \frac{dx}{dt} = f(x) \]

Differential Cut

\[ \frac{dy}{dt} = g(x, y) \]

Differential Ghost

\[ x' = f(x) \]

Logic

Provability theory

Math

Characteristic PDE

JLogComput’10, CAV’08, FMSD’09, LMCS’12, LICS’12, ITP’12, CADE’15

André Platzer (CMU)

FCPS / 22: Axioms & Uniform Substitutions
Differential Invariants for Differential Equations

Differential Invariant

Differential Cut

Differential Ghost

Logic

 Provability theory

Math

 Characteristic PDE

JLogComput'10, CAV'08, FMSD'09, LMCS'12, LICS'12, ITP'12, CADE'15

André Platzer (CMU)
Differential Invariants for Differential Equations

Differential Invariant

Differential Cut

Differential Ghost

\[ \frac{dx}{dt} = f(x) \]

\[ \frac{dy}{dt} = g(x, y) \]

\[ \text{DI} \geq \text{DI} >, \land, \lor \]

\[ \text{DI} = \text{DI} >, =, \land, \lor \]

Logic

Provability theory

Math

Characteristic PDE

JLogComput'10, CAV'08, FMSD'09, LMCS'12, LICS'12, ITP'12, CADE'15

André Platzer (CMU)
Differential Invariants for Differential Equations

Differential Invariant

Differential Cut

Differential Ghost

\[
x' = f(x) \quad 0 \leq t
\]

Logic
Provability theory

Math
Characteristic PDE

JLogComput'10, CAV'08, FMSD'09, LMCS'12, LICS'12, ITP'12, CADE'15
Differential Invariants for Differential Equations

Differential Invariant

Differential Cut

Differential Ghost

Logic

Math

Provability theory

Characteristic PDE

JLogComput’10, CAV’08, FMSD’09, LMCS’12, LICS’12, ITP’12, CADE’15

André Platzer (CMU)
Differential Invariants for Differential Equations

Differential Invariant

Differential Cut

Differential Ghost

Logic
Provability theory

Math
Characteristic PDE

JLogComput’10, CAV’08, FMSD’09, LMCS’12, LICS’12, ITP’12, CADE’15

André Platzer (CMU)

FCPS / 22: Axioms & Uniform Substitutions
Differential Invariants for Differential Equations

Differential Invariant

Differential Cut

Differential Ghost

Logic

Provability theory

Math

Characteristic PDE

JLogComput’10, CAV’08, FMSD’09, LMCS’12, LICS’12, ITP’12, CADE’15

André Platzer (CMU)

FCPS / 22: Axioms & Uniform Substitutions
Differential Invariants for Differential Equations

Differential Invariant

Differential Cut

Differential Ghost

JLogComput'10, CAV'08, FMSD'09, LMCS'12, LICS'12, ITP'12, CADE'15

Logic

Provability theory

Math

Characteristic PDE

x' = f(x)

0

t

x

DI ≥

DI =

DI >

DI ≥, ∧, ∨

DI =, ∧, ∨

DI >, ∧, ∨

DI ≥,

=, ∧, ∨

DI >, =, ∧, ∨

JLogComput'10, CAV'08, FMSD'09, LMCS'12, LICS'12, ITP'12, CADE'15

Andre Platzer (CMU)
Differential Invariants for Differential Equations

Differential Invariant

Differential Cut

Differential Ghost

\[ y' = g(x, y) \]

\[ x' = f(x) \]

Logic

Provability theory

Math

Characteristic PDE

JLogComput'10, CAV'08, FMSD'09, LMCS'12, LICS'12, ITP'12, CADE'15

André Platzer (CMU)
Differential Invariants for Differential Equations

Differential Invariant

$$\begin{align*}
0 & \quad t \\
\mathbf{x} & \quad \mathbf{x}' = f(x) \\
y' & = g(x,y) \\
\text{inv} & \\
x' & = f(x)
\end{align*}$$

Differential Cut

$$\begin{align*}
\text{DI} & > \text{DI}, \land, \lor \\
\text{DI} & \geq \text{DI}, \land, \lor \\
\text{DI} & \geq \text{DI} =, \land, \lor \\
\text{DI} & > \text{DI} =, \land, \lor
\end{align*}$$

Differential Ghost

Logic

Provability theory

Math

Characteristic PDE

JLogComput'10, CAV'08, FMSD'09, LMCS'12, LICS'12, ITP'12, CADE'15

André Platzer (CMU)
Differential equations cannot leave their evolution domains. Implies:

\[ [x' = f(x) & q(x)]p(x) \leftrightarrow [x' = f(x) & q(x)](q(x) \rightarrow p(x)) \]
### Axiom (Differential Cut)  
(CADE’15)

\[
\text{DC} \quad \left( [x' = f(x) \& q(x)]p(x) \leftrightarrow [x' = f(x) \& q(x) \& r(x)]p(x) \right) \leftarrow [x' = f(x) \& q(x)]r(x)
\]

DC is a cut for differential equations.
DC is a differential modal modus ponens K.
Can’t leave \( r(x) \), then might as well restrict state space to \( r(x) \).
Axiom (Differential Cut) (CADE’15)

\[
\text{DC} \quad (\exists [x' = f(x) \& q(x)]p(x) \iff [x' = f(x) \& q(x) \& r(x)]p(x)) \leftarrow [x' = f(x) \& q(x)]r(x)
\]

DC is a cut for differential equations.

DC is a differential modal modus ponens K.

Can’t leave \(r(x)\), then might as well restrict state space to \(r(x)\).
Axiom (Differential Cut) (CADE’15)

\[
\text{DC} \quad \quad (\left[ x' = f(x) \& q(x) \right] p(x) \leftrightarrow \left[ x' = f(x) \& q(x) \land r(x) \right] p(x)) \\
\leftrightarrow \left[ x' = f(x) \& q(x) \right] r(x)
\]

DC is a cut for differential equations.
DC is a differential modal modus ponens K.
Can’t leave \( r(x) \), then might as well restrict state space to \( r(x) \).
**Axiom (Differential Cut)**

\[
\begin{align*}
\text{DC} & : \quad \left[ x' = f(x) \land q(x) \right] p(x) & \iff & \left[ x' = f(x) \land q(x) \land r(x) \right] p(x) \\
& \iff \left[ x' = f(x) \land q(x) \right] r(x)
\end{align*}
\]

DC is a cut for differential equations.

DC is a differential modal modus ponens K.

Can’t leave \( r(x) \), then might as well restrict state space to \( r(x) \).
**Axiom (Differential Cut)** (CADE’15)

\[
\begin{align*}
\text{DC} & \quad ([x' = f(x) \& q(x)] p(x) \iff [x' = f(x) \& q(x) \land r(x)] p(x)) \\
& \quad \iff [x' = f(x) \& q(x)] r(x)
\end{align*}
\]

DC is a cut for differential equations.
DC is a differential modal modus ponens K.
Can’t leave \( r(x) \), then might as well restrict state space to \( r(x) \).
Differential Equation Axioms

**Axiom (Differential Cut)**

\[
\begin{align*}
\text{DC} & \quad \left( [x' = f(x) \& q(x)]p(x) \leftrightarrow [x' = f(x) \& q(x) \& r(x)]p(x) \right) \\
& \quad \leftarrow [x' = f(x) \& q(x)]r(x)
\end{align*}
\]

DC is a cut for differential equations.
DC is a differential modal modus ponens K.
Can’t leave \( r(x) \), then might as well restrict state space to \( r(x) \).
Axiom (Differential Cut) (CADE’15)

$$\begin{align*}
\text{DC} & \quad ( [x' = f(x) \& q(x)] p(x) \iff [x' = f(x) \& q(x) \& r(x)] p(x) ) \\
& \iff [x' = f(x) \& q(x)] r(x)
\end{align*}$$

DC is a cut for differential equations.
DC is a differential modal modus ponens K.
Can’t leave $r(x)$, then might as well restrict state space to $r(x)$. 

Andre Platzer (CMU)
Axiom (Differential Cut) (CADE'15)

\[ DC \left( \left[ x' = f(x) \land q(x) \right] p(x) \leftrightarrow \left[ x' = f(x) \land q(x) \land r(x) \right] p(x) \right) \]

\[ \leftarrow \left[ x' = f(x) \land q(x) \right] r(x) \]

DC is a cut for differential equations.
DC is a differential modal modus ponens K.
Can’t leave \( r(x) \), then might as well restrict state space to \( r(x) \).
Axiom (Differential Cut) (CADE’15)

\[
\text{DC} \quad ([x' = f(x) \& q(x)]p(x) \leftrightarrow [x' = f(x) \& q(x) \& r(x)]p(x))
\leftarrow [x' = f(x) \& q(x)]r(x)
\]

DC is a cut for differential equations.
DC is a differential modal modus ponens K.
Can’t leave \(r(x)\), then might as well restrict state space to \(r(x)\).
Differential Equation Axioms

Axiom (Differential Invariant) (CADE’15)

DI \[ x' = f(x) & q(x) \] \[ p(x) \leftarrow (q(x) \rightarrow p(x) \land [x' = f(x) & q(x)](p(x))') \]

Differential invariant: \( p(x) \) true now and its differential \((p(x))'\) true always

What’s the differential of a formula???
What’s the meaning of a differential term ... in a state???
Axiom (Differential Effect) \[(x' = f(x) & q(x)]p(x, x') \leftrightarrow [x' = f(x) & q(x)][x' := f(x)]p(x, x')\]

Effect of differential equation on differential symbol $x'$:

- $[x' := f(x)]$ instantly mimics continuous effect $[x' = f(x)]$ on $x'$.
- $[x' := f(x)]$ selects vector field $x' = f(x)$ for subsequent differentials.
Differential Equation Axioms

### Axiom (Differential Ghost) (CADE’15)

| Differential Ghost/auxiliaries: extra differential equations that exist |
| Can cause new invariants |
| “Dark matter” counterweight to balance conserved quantities |

\[
\text{DG} \quad [x' = f(x) \& q(x)] p(x) \iff \exists y [x' = f(x), y' = a(x)y + b(x) \& q(x)] p(x)
\]
Differential solutions: solve differential equations with DG,DC and inverse companions

Differential Equation Axioms

Axiom (Differential Solution) (CADE’15)

\[ DS \ [x' = f & q(x)]p(x) \iff \forall t \geq 0 \ ((\forall 0 \leq s \leq t \ q(x+fs)) \rightarrow [x := x+ft]p(x)) \]
Example: Differential Invariants Don’t Solve. Prove!

Differential Invariants (DI) prove properties of an ODE inductively by its differentials. DI exports a vector field, possibly after Dynamic Window (DW) exports the evolution domain. DI can reason efficiently in Equivalence or EQuational context. DI isolates the postcondition.

\[ x \cdot x \geq 1 \rightarrow [x' = x^3] x \cdot x \geq 1 \]
**Example: Differential Invariants Don’t Solve. Prove!**

1. DI proves a property of an ODE inductively by its differentials

\[
(x' = x^3)(x \cdot x \geq 1)'
\]

\[
x \cdot x \geq 1 \rightarrow [x' = x^3]x \cdot x \geq 1
\]
1. DI proves a property of an ODE inductively by its的不同\solutions\s\s\n2. DE exports vector field, possibly after DW exports evolution domain

\[\begin{align*}
\text{CE} & \quad [x' = x^3][x' := x^3](x \cdot x \geq 1)' \\
\text{DE} & \quad [x' = x^3](x \cdot x \geq 1)'
\end{align*}\]

\[x \cdot x \geq 1 \rightarrow [x' = x^3]x \cdot x \geq 1\]
DI proves a property of an ODE inductively by its differentials

DE exports vector field, possibly after DW exports evolution domain

CE+CQ reason efficiently in Equivalence or eQuational context

\[
\begin{align*}
G & \quad [x' = x^3][x' := x^3]x' \cdot x + x \cdot x' \geq 0 \\
CE & \quad (x \cdot x \geq 1)' \iff x' \cdot x + x \cdot x' \geq 0 \\
DE & \quad [x' = x^3][x' := x^3](x \cdot x \geq 1)' \\
DI & \quad x \cdot x \geq 1 \rightarrow [x' = x^3]x \cdot x \geq 1
\end{align*}
\]
DI proves a property of an ODE inductively by its differentials

DE exports vector field, possibly after DW exports evolution domain

CE+CQ reason efficiently in Equivalence or eQuational context

G isolates postcondition

\[
\begin{align*}
\text{CE} & \quad [\dot{x'} = x^3][\dot{x} := x^3] (x \cdot x \geq 1)' \\
\text{DE} & \quad [\dot{x'} = x^3] (x \cdot x \geq 1)'
\end{align*}
\]

\[
\begin{align*}
\text{DI} & \quad x \cdot x \geq 1 \rightarrow [\dot{x'} = x^3] x \cdot x \geq 1
\end{align*}
\]
1. DI proves a property of an ODE inductively by its differentials.
2. DE exports vector field, possibly after DW exports evolution domain.
3. CE+CQ reason efficiently in Equivalence or eQuational context.
5. \([\cdot :=]\) differential substitution uses vector field.

\[
\begin{align*}
\text{DE} & \quad \left[ x' := x^3 \right] x' \cdot x + x \cdot x' \geq 0 \\
\text{CE} & \quad \left[ x' := x^3 \right] \left[ x' := x^3 \right] x' \cdot x + x \cdot x' \geq 0 \\
\text{G} & \quad \left[ x' = x^3 \right] \left[ x' := x^3 \right] (x \cdot x \geq 1)' \\
\text{DI} & \quad x \cdot x \geq 1 \rightarrow \left[ x' = x^3 \right] x \cdot x \geq 1
\end{align*}
\]
Example: Differential Invariants Don’t Solve. Prove!

1. DI proves a property of an ODE inductively by its differentials
2. DE exports vector field, possibly after DW exports evolution domain
3. CE+CQ reason efficiently in Equivalence or eQuational context
4. G isolates postcondition
5. [′:=] differential substitution uses vector field

\[
\begin{align*}
\mathbb{R} & \quad x^3 \cdot x + x \cdot x^3 \geq 0 \\
[′:=] & \quad [x′ := x^3]x′ \cdot x + x \cdot x′ \geq 0 \\
G & \quad [x′ = x^3][x′ := x^3]x′ \cdot x + x \cdot x′ \geq 0 \\
\quad & \quad (x \cdot x \geq 1)′ \iff x′ \cdot x + x \cdot x′ \geq 0 \\
CE & \quad [x′ = x^3][x′ := x^3](x \cdot x \geq 1)′ \\
DE & \quad [x′ = x^3](x \cdot x \geq 1)′ \\
DI & \quad x \cdot x \geq 1 \rightarrow [x′ = x^3]x \cdot x \geq 1
\end{align*}
\]
## Example: Differential Invariants Don't Solve. Prove!

1. **DI** proves a property of an ODE inductively by its differentials
2. **DE** exports vector field, possibly after **DW** exports evolution domain
3. **CE+CQ** reason efficiently in Equivalence or eQuational context
4. **G** isolates postcondition
5. **[':=]** differential substitution uses vector field

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>R</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>G</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image2.png" alt="Image" /></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CE</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image3.png" alt="Image" /></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DE</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image4.png" alt="Image" /></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DI</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image5.png" alt="Image" /></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: Differential Invariants Don’t Solve. Prove!

1. DI proves a property of an ODE inductively by its differentials
2. DE exports vector field, possibly after DW exports evolution domain
3. CE+CQ reason efficiently in Equivalence or eQuational context
4. G isolates postcondition
5. [$':=\)$ differential substitution uses vector field

\[ x^3 \cdot x + x \cdot x^3 \geq 0 \]

\[ \text{DE} [x' := x^3][x' := x^3] x' \cdot x + x \cdot x' \geq 0 \]

\[ \text{CE} [x' = x^3][x' := x^3] x' \cdot x + x \cdot x' \geq 0 \]

\[ (x \cdot x)' = x' \cdot x + x \cdot x' \]

\[ (x \cdot x)' \geq 0 \iff x' \cdot x + x \cdot x' \geq 0 \]

\[ (x \cdot x \geq 1)' \iff x' \cdot x + x \cdot x' \geq 0 \]

\[ x \cdot x \geq 1 \rightarrow [x' = x^3] x \cdot x \geq 1 \]
1. DI proves a property of an ODE inductively by its differentials.
2. DE exports vector field, possibly after DW exports evolution domain.
3. CE+CQ reason efficiently in Equivalence or eQuational context.
5. ['='] differential substitution uses vector field.

\[
\begin{align*}
\mathbb{R} & \quad x^3 \cdot x + x \cdot x^3 \geq 0 \\
['] & = \quad [x' := x^3] x' \cdot x + x \cdot x' \geq 0 \\
G & \quad [x' = x^3][x' := x^3] x' \cdot x + x \cdot x' \geq 0 \\
CE & \quad [x' = x^3][x' := x^3](x \cdot x \geq 1)' \\
DE & \quad [x' = x^3](x \cdot x \geq 1)' \\
DI & \quad x \cdot x \geq 1 \rightarrow [x' = x^3]x \cdot x \geq 1
\end{align*}
\]
DI proves a property of an ODE inductively by its differentials
DE exports vector field, possibly after DW exports evolution domain
CE+CQ reason efficiently in Equivalence or eQuational context
G isolates postcondition
[′:=] differential substitution uses vector field
· differential computations are axiomatic (US)

\[
\begin{align*}
\text{DI} & \quad x \cdot x + x \cdot x^3 \geq 0 \\
\text{DE} & \quad x' = x^3 & x' \cdot x + x \cdot x' \geq 0 \\
\text{CE} & \quad [x' = x^3][x' := x^3]x' \cdot x + x \cdot x' \geq 0 \\
\text{CQ} & \quad (x \cdot x)' \geq 0 \iff x' \cdot x + x \cdot x' \geq 0 \\
\text{US} & \quad (f(\bar{x}) \cdot g(\bar{x}))' = (f(\bar{x}))' \cdot g(\bar{x}) + f(\bar{x}) \cdot (g(\bar{x}))' \\
\text{DI} & \quad x \cdot x \geq 1 \rightarrow [x' = x^3]x \cdot x \geq 1
\end{align*}
\]
Example: Differential Invariants Don’t Solve. Prove!

1. DI proves a property of an ODE inductively by its differentials
2. DE exports vector field, possibly after DW exports evolution domain
3. CE+CQ reason efficiently in Equivalence or eQuational context
4. G isolates postcondition
5. ':= differential substitution uses vector field
6. ·' differential computations are axiomatic (US)

\[
\begin{align*}
\mathbb{R} & : x^3 \cdot x + x \cdot x^3 \geq 0 \\
\mathbb{G} & : [x' := x^3] x' \cdot x + x \cdot x' \geq 0 \\
\mathbb{C} & : [x' = x^3] [x' := x^3] x' \cdot x + x \cdot x' \geq 0 \\
\mathbb{E} & : [x' = x^3] (x \cdot x \geq 1)' \\
\mathbb{D} & : [x' = x^3] (x \cdot x \geq 1)' \\
\end{align*}
\]

\[
\begin{align*}
\mathbb{G} \quad & x \cdot x \geq 1 \rightarrow [x' = x^3] x \cdot x \geq 1 \\
\mathbb{D} \quad & x \cdot x \geq 1 \rightarrow [x' = x^3] x \cdot x \geq 1 \\
\end{align*}
\]

\[
\begin{align*}
\mathbb{U} & : (f(\bar{x}) \cdot g(\bar{x}))' = (f(\bar{x}))(g(\bar{x})' + f(\bar{x}) \cdot (g(\bar{x}))') \\
\mathbb{C} & : (x \cdot x)' = (x)' \cdot x + x \cdot (x)' \\
\end{align*}
\]
Example: Soundly Solving Differential Equations

1. DG introduces time $t$, DC cuts solution in, that DI proves and
2. DW exports to postcondition
3. inverse DC removes evolution domain constraints
4. inverse DG removes original ODE
5. DS solves remaining ODE for time

\[
\phi \rightarrow \forall s \geq 0 (x_0 + \frac{a}{2} s^2 + v_0 s \geq 0)
\]

\[
[=] \phi \rightarrow \forall s \geq 0 [t := 0 + 1s] x_0 + \frac{a}{2} t^2 + v_0 t \geq 0
\]

\[
\text{DS} \phi \rightarrow [t' = 1] x_0 + \frac{a}{2} t^2 + v_0 t \geq 0
\]

\[
\text{DG} \phi \rightarrow [v' = a, t' = 1] x_0 + \frac{a}{2} t^2 + v_0 t \geq 0
\]

\[
\text{DG} \phi \rightarrow [x' = v, v' = a, t' = 1] x_0 + \frac{a}{2} t^2 + v_0 t \geq 0
\]

\[
\text{DC} \phi \rightarrow [x' = v, v' = a, t' = 1 & v = v_0 + at] x_0 + \frac{a}{2} t^2 + v_0 t \geq 0
\]

\[
\text{DC} \phi \rightarrow [x' = v, v' = a, t' = 1 & v = v_0 + at \land x = x_0 + \frac{a}{2} t^2 + v_0 t] x_0 + \frac{a}{2} t^2 + v_0 t \geq 0
\]

\[
\text{G,K} \phi \rightarrow [x' = v, v' = a, t' = 1 & v = v_0 + at \land x = x_0 + \frac{a}{2} t^2 + v_0 t] (x = x_0 + \frac{a}{2} t^2 + v_0 t \rightarrow x \geq 0)
\]

\[
\text{DW} \phi \rightarrow [x' = v, v' = a, t' = 1 & v = v_0 + at \land x = x_0 + \frac{a}{2} t^2 + v_0 t] x \geq 0
\]

\[
\text{DC} \phi \rightarrow [x' = v, v' = a, t' = 1 & v = v_0 + at] x \geq 0
\]

\[
\text{DC} \phi \rightarrow [x' = v, v' = a, t' = 1] x \geq 0
\]

\[
\phi \rightarrow \exists t [x' = v, v' = a, t' = 1] x \geq 0
\]

\[
\text{DG} \phi \rightarrow [x' = v, v' = a] x \geq 0
\]
The Meaning of Prime

Differential Forms

\[ \theta' \]

\[ \sum x_u(x') \frac{\partial [\theta]}{\partial x^i(u)} \]

depends on the differential equation?

well-defined in isolated state

\[ u \]

André Platzer (CMU)
Semantics

\[ \llbracket (\theta)' \rrbracket l u = \]
\( [(\theta)'] I u = \)

depends on the differential equation?
Semantics \( [(\theta)' \iota u = \ldots ] \)

depends on the differential equation?
well-defined in isolated state \( u \) at all?
Semantics \[ (\theta')lu = \sum_x u(x') \frac{\partial[\theta]}{\partial x}(u) \]

depends on the differential equation?

well-defined in isolated state \( u \) at all?
Semantics: \[ I((θ')')lu = \sum_x u(x') \frac{∂[θ]}{∂x}(u) \]

depends on the differential equation? well-defined in isolated state \( u \) at all?
Lemma (Differential lemma)

If \( I, \varphi \models x' = f(x) \land Q \) for duration \( r > 0 \), then for all \( 0 \leq \zeta \leq r \):

\[
[(\theta)']I\varphi(\zeta) = \frac{d[\theta]I\varphi(t)}{dt}(\zeta)
\]

Syntactic \( \Rightarrow \) Analytic

Lemma (Differential assignment)

If \( I, \varphi \models x' = f(x) \land Q \) then \( I, \varphi \models \phi \leftrightarrow [x' := f(x)]\phi \)

Lemma (Derivations)

\[
(f(\bar{x}) + g(\bar{x}))' = (f(\bar{x}))' + (g(\bar{x}))'
\]

\[
(f(\bar{x}) \cdot g(\bar{x}))' = (f(\bar{x}))' \cdot g(\bar{x}) + f(\bar{x}) \cdot (g(\bar{x}))'
\]

\[
[y := f(\bar{x})][y' := 1]((f(f(\bar{x})))' = (f(y))' \cdot (f(\bar{x})))'
\]

\[
(f)' = 0
\]

for \( y, y' \not\in f(\bar{x}) \)

for arity 0 functions/numbers
CPS are Multi-Dynamical Systems

Uniform Substitution Calculus for Differential Dynamic Logic
- Uniform Substitution Calculus
- Axiom vs. Axiom Schema
- Uniform Substitutions
- Uniform Substitution Lemmas
- Differential Axioms
- Differential Invariants
- Examples

Differential-form Differential Dynamic Logic
- Syntax
- Semantics
- Differential Substitution Lemmas
- Contextual Congruences
- Parametric Computational Proofs
- Static Semantics

Summary
Differential Equation Axioms

Axiom (Differential Invariant) (CADE’15)

\[ [x' = f(x) \& q(x)]p(x) \iff (q(x) \to p(x) \& [x' = f(x) \& q(x)](p(x)))' \]

Differential invariant: \( p(x) \) true now and its differential \( (p(x))' \) true always

What’s the differential of a formula???

What’s the meaning of a differential term . . . in a state???
### Definition (Hybrid program $\alpha$)

\[
a | x := \theta | x' := \theta | ?Q | x' = f(x) \& Q | \alpha \cup \beta | \alpha; \beta | \alpha^*
\]

### Definition (dL Formula $\phi$)

\[
\theta \geq \eta | p(\theta_1, \ldots, \theta_k) | \neg \phi | \phi \land \psi | \forall x \phi | \exists x \phi | [\alpha] \phi | \langle \alpha \rangle \phi
\]

### Definition (Term $\theta$)

\[
x | x' | f(\theta_1, \ldots, \theta_k) | \theta + \eta | \theta \cdot \eta | (\theta)'\]
### Differential-form Differential Dynamic Logic: Syntax

#### Definition (Hybrid program $\alpha$)

$$
\begin{align*}
& a \mid \mathit{x} := \theta \mid \mathit{x}' := \theta \mid ?Q \mid \mathit{x}' = f(x) \& Q \\
& \alpha \cup \beta \mid \alpha; \beta \mid \alpha^* 
\end{align*}
$$

#### Definition (dL Formula $\phi$)

$$
\begin{align*}
& \theta \geq \eta \mid p(\theta_1, \ldots, \theta_k) \mid \neg \phi \mid \phi \land \psi \mid \forall x \phi \mid \exists x \phi \mid [\alpha] \phi \mid \langle \alpha \rangle \phi 
\end{align*}
$$

#### Definition (Term $\theta$)

$$
\begin{align*}
& \mathit{x} \mid \mathit{x}' \mid f(\theta_1, \ldots, \theta_k) \mid \theta + \eta \mid \theta \cdot \eta \mid (\theta)' 
\end{align*}
$$
Differential-form Differential Dynamic Logic: Syntax

**Definition (Hybrid program $\alpha$)**

\[
a | x := \theta | x' := \theta | ?Q | x' = f(x) & Q | \alpha \cup \beta | \alpha; \beta | \alpha^*
\]

**Definition (dL Formula $\phi$)**

\[
\theta \geq \eta | p(\theta_1, \ldots, \theta_k) | \neg \phi | \phi \land \psi | \forall x \phi | \exists x \phi | [\alpha] \phi | \langle \alpha \rangle \phi
\]

**Definition (Term $\theta$)**

\[
x | x' | f(\theta_1, \ldots, \theta_k) | \theta + \eta | \theta \cdot \eta | (\theta)'\]
**Definition (Term semantics)**

\[
[(\theta)']_I u = \sum_x u(x') \frac{\partial [\theta]\_I}{\partial x}(u) = \sum_x u(x') \frac{\partial [\theta]_I u^X_x}{\partial X}
\]

**Definition (dL semantics)**

\[
[(C(\phi))\_I] = I(C)([\phi]_I)
\]
\[
[(\langle \alpha \rangle \phi)\_I] = [\alpha]_I \circ [\phi]_I
\]
\[
[[\alpha] \phi]_I = [\neg \langle \alpha \rangle \neg \phi]_I
\]

**Definition (Program semantics)**

\[
[x' = f(x) \& Q]_I = \{(\varphi(0)|\{x',c\}, \varphi(r)) : I, \varphi \models x' = f(x) \land Q\}
\]
\[
[\alpha \cup \beta]_I = [\alpha]_I \cup [\beta]_I
\]
\[
[\alpha; \beta]_I = [\alpha]_I \circ [\beta]_I
\]
\[
[\alpha^*]_I = ([\alpha]_I)^* = \bigcup_{n \in \mathbb{N}} [\alpha^n]_I
\]
Differential Substitution Lemmas

**Lemma (Differential lemma)**

If $I, \varphi \models x' = f(x) \land Q$ for duration $r > 0$, then for all $0 \leq \zeta \leq r$:

$$[\eta]' I \varphi(\zeta) = \frac{d[\eta] I \varphi(t)}{dt}(\zeta)$$

**Lemma (Differential assignment)**

If $I, \varphi \models x' = f(x) \land Q$ then $I, \varphi \models \phi \leftrightarrow [x' := f(x)]\phi$

**Lemma (Derivations)**

$$(\theta + \eta)' = (\theta)' + (\eta)'$$

$$(\theta \cdot \eta)' = (\theta)' \cdot \eta + \theta \cdot (\eta)'$$

$$[y := \theta][y' := 1]((f(\theta))' = (f(y))' \cdot (\theta)')$$

for $y, y' \notin \theta$

$$(f)' = 0$$

for arity 0 functions/numbers $f$
Differential Equation Axioms & Differential Axioms

\[ x' = f(x) \& q(x) \]

\[ p(x) \leftrightarrow [x' = f(x) \& q(x)]p(x) \]

\[ [x' = f(x) \& q(x)]p(x) \leftrightarrow [x' = f(x) \& q(x) \wedge r(x)]p(x) \]

\[ \leftarrow [x' = f(x) \& q(x)]r(x) \]

\[ [x' = f(x) \& q(x)]p(x) \leftrightarrow [x' = f(x) \& q(x)]p(x, x') \]

\[ q(x) \rightarrow p(x) \wedge [x' = f(x) \& q(x)](p(x))' \]

\[ [x' = f(x) \& q(x)]p(x) \leftrightarrow \exists y [x' = f(x), y' = a(x)y + b(x) \& q(x)]p(x) \]

\[ \forall t \geq 0 ((\forall 0 \leq s \leq t q(x + fs)) \rightarrow [x := x + ft]p(x)) \]

\[ [x' := f]p(x') \leftrightarrow p(f) \]

\[ (f(x) + g(x))' = (f(x))' + (g(x))' \]

\[ (f(x) \cdot g(x))' = (f(x))' \cdot g(x) + f(x) \cdot (g(x))' \]

\[ [y := g(x)][y' := 1](f(g(x)))' = (f(y))' \cdot (g(x))' \]
Differential Dynamic Logic: Axioms

\[
\begin{align*}
G & \quad \frac{p(\bar{x})}{[a]p(\bar{x})} \\
\forall & \quad \frac{p(x)}{\forall x \ p(x)} \\
\text{MP} & \quad \frac{p \rightarrow q \quad p}{q} \\
\text{CT} & \quad \frac{f(\bar{x}) = g(\bar{x})}{c(f(\bar{x})) = c(g(\bar{x}))} \\
\text{CQ} & \quad \frac{f(\bar{x}) = g(\bar{x})}{p(f(\bar{x})) \leftrightarrow p(g(\bar{x}))} \\
\text{CE} & \quad \frac{p(\bar{x}) \leftrightarrow q(\bar{x})}{C(p(\bar{x})) \leftrightarrow C(q(\bar{x}))}
\end{align*}
\]
1. **DI** proves a property of an ODE inductively by its differentials.
2. **DE** exports vector field, possibly after **DW** exports evolution domain.
3. **CE+CQ** reason efficiently in Equivalence or eQuational context.
4. **G** isolates postcondition.
5. `[':=]` differential substitution uses vector field.

<table>
<thead>
<tr>
<th>Step</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DI</strong></td>
<td>$x^3 \cdot x + x \cdot x^3 \geq 0$</td>
</tr>
<tr>
<td><strong>DE</strong></td>
<td>$(x \cdot x)' = x' \cdot x + x \cdot x'$</td>
</tr>
<tr>
<td><strong>CE</strong></td>
<td>$(x \cdot x)' \geq 0 \iff x' \cdot x + x \cdot x' \geq 0$</td>
</tr>
<tr>
<td><strong>G</strong></td>
<td>$(x \cdot x \geq 1)' \iff x' \cdot x + x \cdot x' \geq 0$</td>
</tr>
<tr>
<td><strong>CQ</strong></td>
<td>$(x \cdot x \geq 1)'$</td>
</tr>
</tbody>
</table>

\[ x \cdot x \geq 1 \rightarrow [x' = x^3] x \cdot x \geq 1 \]
Example: Syntactic Contextual Congruence by US

\[
\begin{align*}
\text{CQ} & \quad f(\bar{x}) = g(\bar{x}) \\
& \quad p(f(\bar{x})) \leftrightarrow p(g(\bar{x})) \\
\text{CQ} & \quad (x \cdot x)' = x' \cdot x + x \cdot x' \\
& \quad (x \cdot x)' \geq 0 \leftrightarrow x' \cdot x + x \cdot x' \geq 0 \\
\text{CE} & \quad p(\bar{x}) \leftrightarrow q(\bar{x}) \\
& \quad C(p(\bar{x})) \leftrightarrow C(q(\bar{x})) \\
\text{CE} & \quad (x \cdot x \geq 1)' \leftrightarrow x' \cdot x + x \cdot x' \geq 0 \\
& \quad [x' = x^3][x' := x^3](x \cdot x \geq 1)' \leftrightarrow [x' = x^3][x' := x^3]x' \cdot x + x \cdot x' \geq 0
\end{align*}
\]
Example: Syntactic Contextual Congruence by US

\[
\begin{align*}
\text{CQ} & \quad f(\bar{x}) = g(\bar{x}) \\
& \quad p(f(\bar{x})) \leftrightarrow p(g(\bar{x}))
\end{align*}
\]

\[
\begin{align*}
\text{CQ} & \quad (x \cdot x)' = x' \cdot x + x \cdot x' \\
& \quad (x \cdot x)' \geq 0 \iff x' \cdot x + x \cdot x' \geq 0
\end{align*}
\]

with \( \sigma \approx p(\cdot) \mapsto \cdot \geq 0, f(\cdot) \mapsto ((\cdot) \cdot (\cdot))', g(\cdot) \mapsto (\cdot') \cdot (\cdot) + (\cdot) \cdot (\cdot') \)

\[
\begin{align*}
\text{CE} & \quad p(\bar{x}) \leftrightarrow q(\bar{x}) \\
& \quad C(p(\bar{x})) \leftrightarrow C(q(\bar{x}))
\end{align*}
\]

\[
\begin{align*}
\text{CE} & \quad (x \cdot x \geq 1)' \iff x' \cdot x + x \cdot x' \geq 0 \\
& \quad [x' = x^3][x' := x^3](x \cdot x \geq 1)' \iff [x' = x^3][x' := x^3]x' \cdot x + x \cdot x' \geq 0
\end{align*}
\]

with

\[
\begin{align*}
\sigma \approx C(\_)[x' = x^3][x' := x^3], p(\bar{x}) \mapsto ((\cdot)(\cdot) \geq 1)', q(\bar{x}) \mapsto \cdot' + \cdots'
\end{align*}
\]
Example: Syntactic Contextual Congruence by US

\[
\begin{align*}
\text{CQ} & \quad \frac{f(\bar{x}) = g(\bar{x})}{p(f(\bar{x})) \leftrightarrow p(g(\bar{x}))} \\
\text{CQ} & \quad \frac{(x \cdot x)' = x' \cdot x + x \cdot x'}{(x \cdot x)' \geq 0 \iff x' \cdot x + x \cdot x' \geq 0}
\end{align*}
\]

with \( \sigma \approx p(\cdot) \mapsto \cdot \geq 0, f(\bar{x}) \mapsto (x \cdot x)', g(\bar{x}) \mapsto x' \cdot x + x \cdot x' \)

\[
\begin{align*}
\text{CE} & \quad \frac{p(\bar{x}) \leftrightarrow q(\bar{x})}{C(p(\bar{x})) \leftrightarrow C(q(\bar{x}))} \\
\text{CE} & \quad \frac{(x \cdot x \geq 1)' \leftrightarrow x' \cdot x + x \cdot x' \geq 0}{[x' = x^3][x' := x^3](x \cdot x \geq 1)' \leftrightarrow [x' = x^3][x' := x^3]x' \cdot x + x \cdot x' \geq 0}
\end{align*}
\]

with \( \sigma \approx C(\_ \_ ) \mapsto [x' = x^3]p(\bar{x}) \mapsto (x \cdot x \geq 1)', q(\bar{x}) \mapsto x' \cdot x + x \cdot x' \geq 0 \)
<table>
<thead>
<tr>
<th>CE</th>
<th>$[x' = x^3][x' := x^3](x \cdot x \geq 1)'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>DE</td>
<td>$[x' = x^3](x \cdot x \geq 1)'$</td>
</tr>
<tr>
<td>DI</td>
<td>$x \cdot x \geq 1 \rightarrow [x' = x^3]x \cdot x \geq 1$</td>
</tr>
</tbody>
</table>
1 Free function $j(x, x')$ for parametric differential computation

<table>
<thead>
<tr>
<th>Step</th>
<th>Rule</th>
<th>Assertion</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>$[x' = x^3][x' := x^3] j(x, x') \geq 0$</td>
<td>$(x \cdot x \geq 1)' \iff j(x, x') \geq 0$</td>
</tr>
<tr>
<td>CE</td>
<td>$[x' = x^3][x' := x^3] (x \cdot x \geq 1)'$</td>
<td></td>
</tr>
<tr>
<td>DE</td>
<td>$[x' = x^3] (x \cdot x \geq 1)'$</td>
<td></td>
</tr>
<tr>
<td>DI</td>
<td>$x \cdot x \geq 1 \rightarrow [x' = x^3] x \cdot x \geq 1$</td>
<td></td>
</tr>
</tbody>
</table>
Free function $j(x, x')$ for parametric differential computation

Again $G,[':=]$ to isolate differentially substituted postcondition

\[
\begin{align*}
\text{CE} & \quad [x' = x^3][x' := x^3]j(x, x') \geq 0 \\
\text{DE} & \quad [x' = x^3][x' := x^3](x \cdot x \geq 1)' \\
\text{DI} & \quad x \cdot x \geq 1 \rightarrow [x' = x^3]x \cdot x \geq 1
\end{align*}
\]
1. Free function $j(x, x')$ for parametric differential computation
2. Again $G,[\cdot :=]$ to isolate differentially substituted postcondition

\[ j(x, x^3) \geq 0 \]

\[
\begin{align*}
\left[ x' := x^3 \right] j(x, x') &\geq 0 \\
G[\cdot :=] &\Rightarrow [x' = x^3][x' := x^3] j(x, x') \geq 0 \\
(x \cdot x \geq 1)' &\leftrightarrow j(x, x') \geq 0 \\
\end{align*}
\]

CE

\[
\begin{align*}
[\cdot :=] &\Rightarrow \left[ x' = x^3 \right][x' := x^3] (x \cdot x \geq 1)' \\
\end{align*}
\]

DE

\[
\begin{align*}
[\cdot :=] &\Rightarrow [x' = x^3](x \cdot x \geq 1)' \\
\end{align*}
\]

DI

\[
\begin{align*}
x \cdot x \geq 1 \rightarrow [x' = x^3]x \cdot x \geq 1 \\
\end{align*}
\]
1. Free function $j(x, x')$ for parametric differential computation
2. Again $G, [':=]$ to isolate differentially substituted postcondition
3. Construct parametric $j(x, x')$ by axiomatic differential computation

\[
\begin{align*}
\text{CE} & \quad \begin{cases} 
  j(x, x^3) \geq 0 \\
  [x' := x^3]j(x, x') \geq 0
\end{cases} \\
\text{DE} & \quad [x' = x^3][x' := x^3]j(x, x') \geq 0 \\
\text{DI} & \quad \begin{cases} 
  (x \cdot x \geq 1)' \iff j(x, x') \geq 0 \\
  j(x, x') \geq 0 \\
  (x \cdot x \geq 1)' \iff j(x, x') \geq 0 \\
  (x \cdot x \geq 1)' \iff j(x, x') \geq 0
\end{cases}
\end{align*}
\]

\[
\begin{align*}
\text{CQ} & \quad (x \cdot x)' \geq 0 \iff j(x, x') \geq 0 \\
\implies & \quad (x \cdot x \geq 1)' \iff j(x, x') \geq 0 \\
\implies & \quad \begin{cases} 
  (x \cdot x \geq 1)' \iff j(x, x') \geq 0 \\
  j(x, x') \geq 0 \\
  (x \cdot x \geq 1)' \iff j(x, x') \geq 0 \\
  x \cdot x \geq 1 \implies [x' = x^3][x \cdot x \geq 1]
\end{cases}
\end{align*}
\]
**Example: Differential Invariants**

1. Free function $j(x, x')$ for parametric differential computation
2. Again $G, ['=:']$ to isolate differentially substituted postcondition
3. Construct parametric $j(x, x')$ by axiomatic differential computation

<table>
<thead>
<tr>
<th>Step</th>
<th>Rule</th>
<th>Hypothesis</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>['] :=</td>
<td>$j(x, x^3) \geq 0$</td>
<td>$x' := x^3 j(x, x') \geq 0$</td>
</tr>
<tr>
<td>2</td>
<td>$G$</td>
<td>$[x' := x^3][x' := x^3] j(x, x') \geq 0$</td>
<td>$(x \cdot x)' = j(x, x')$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$CQ (x \cdot x)' \geq 0 \iff j(x, x') \geq 0$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$(x \cdot x \geq 1)' \iff j(x, x') \geq 0$</td>
</tr>
<tr>
<td>3</td>
<td>CE</td>
<td>$[x' = x^3][x' := x^3](x \cdot x \geq 1)'$</td>
<td>$x' = x^3$ $[x' := x^3](x \cdot x \geq 1)'$</td>
</tr>
<tr>
<td></td>
<td>DE</td>
<td>$[x' = x^3](x \cdot x \geq 1)'$</td>
<td>$x \cdot x \geq 1 \rightarrow [x' = x^3] x \cdot x \geq 1$</td>
</tr>
<tr>
<td></td>
<td>DI</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Free function $j(x, x')$ for parametric differential computation

Again $G, ['':=]$, to isolate differentially substituted postcondition

Construct parametric $j(x, x')$ by axiomatic differential computation

USR instantiates proof by $\{ j(x, x') \leftrightarrow x' \cdot x + x \cdot x' \}$

CQ $j(x, x') \geq 0 \iff (x \cdot x)' = j(x, x')$

CE $j(x, x^3) \geq 0 \vdash x' := x^3 j(x, x') \geq 0$

DE $x' = x^3 \vdash j(x, x') \geq 0 \implies (x \cdot x) \geq 1' \iff j(x, x') \geq 0$

DI $x \cdot x \geq 1 \rightarrow [x' = x^3] x \cdot x \geq 1$

USR $x^3 \cdot x + x \cdot x^3 \geq 0 \vdash x' (x \cdot x)' = x' \cdot x + x \cdot x'$

$\cdot x \geq 1 \rightarrow [x' = x^3] x \cdot x \geq 1$
1. Free function \( j(x, x') \) for parametric differential computation
2. Again \( G, ['=:] \) to isolate differentially substituted postcondition
3. Construct parametric \( j(x, x') \) by axiomatic differential computation
4. USR instantiates proof by \( \{ j(x, x') \mapsto x' \cdot x + x \cdot x' \} \)

\[
\begin{align*}
j(x, x^3) &\geq 0 \\
[j' := [x := x^3]j(x, x')] &\leq 0 \\
[\cdot] &\leq [x' = x^3][x := x^3]j(x, x') \geq 0 \\
G &\leq CQ (x \cdot x)' \geq 0 \iff j(x, x') \geq 0 \\
(x \cdot x )' &\leq (x \cdot x )' \iff j(x, x') \geq 0 \\
\end{align*}
\]

\[
\begin{align*}
CE &\leq \frac{[x' = x^3][x := x^3]j(x, x') \geq 0}{(x \cdot x )' = j(x, x')} \\
DE &\leq x \cdot x \geq 1 \rightarrow [x' = x^3]x \cdot x \geq 1 \\
DI &\leq x \cdot x \geq 1 \rightarrow [x' = x^3]x \cdot x \geq 1 \\
\end{align*}
\]

\[
\begin{align*}
\mathbb{R} &\geq x^3 \cdot x + x \cdot x^3 \geq 0 \\
USR &\leq x \cdot x \geq 1 \rightarrow [x' = x^3]x \cdot x \geq 1 \\
\end{align*}
\]
1. Free function $j(x, x')$ for parametric differential computation
2. Again $G, [':=]$ to isolate differentially substituted postcondition
3. Construct parametric $j(x, x')$ by axiomatic differential computation
4. USR instantiates proof by \{ $j(x, x') \mapsto x' \cdot x + x \cdot x'$ \}

\[ j(x, x^3) \geq 0 \]
\[ [x' := x^3]j(x, x') \geq 0 \]
\[ G \]
\[ [x' = x^3][x' := x^3]j(x, x') \geq 0 \]

\[ j(x, x') \mapsto (x \cdot x)' = j(x, x') \]

\[ (x \cdot x)' \geq 0 \iff j(x, x') \geq 0 \]
\[ (x \cdot x ' \geq 1)' \leftrightarrow j(x, x') \geq 0 \]

\[ CE \]
\[ [x' = x^3][x' := x^3](x \cdot x \geq 1)' \]

\[ DE \]
\[ [x' = x^3](x \cdot x \geq 1)' \]

\[ DI \]
\[ x \cdot x \geq 1 \rightarrow [x' = x^3]x \cdot x \geq 1 \]

\[ \mathbb{R} \]
\[ x^3 \cdot x + x \cdot x^3 \geq 0 \]

\[ USR \]
\[ x \cdot x \geq 1 \rightarrow [x' = x^3]x \cdot x \geq 1 \]

\[ US \]
\[ (x \cdot x)' = (x)' \cdot x + x \cdot (x)' \]
\[ x' \]
\[ (x \cdot x)' = x' \cdot x + x \cdot x' \]
Example: Differential Invariants

1. Free function $j(x, x')$ for parametric differential computation
2. Again $G,[':=]$ to isolate differentially substituted postcondition
3. Construct parametric $j(x, x')$ by axiomatic differential computation
4. USR instantiates proof by $\{j(x, x') \mapsto x' \cdot x + x \cdot x'\}$

\[
\begin{align*}
\text{USR} \quad & \quad j(x, x^3) \geq 0 \\
\hline
\text{G} \quad & \quad [x' := x^3]j(x, x') \geq 0 \\
\hline
\text{CE} \quad & \quad [x' = x^3][x' := x^3](x \cdot x') = j(x, x') \\
\text{DI} \quad & \quad (x \cdot x) \geq 1 \rightarrow [x' = x^3]x \cdot x \geq 1
\end{align*}
\]

\[
\begin{align*}
\text{CQ} \quad & \quad (x \cdot x) \geq 1 \leftrightarrow j(x, x') \geq 0 \\
\end{align*}
\]
1. Free function $j(x, x')$ for parametric differential computation
2. Again $G, ['':=]$ to isolate differentially substituted postcondition
3. Construct parametric $j(x, x')$ by axiomatic differential computation
4. USR instantiates proof by \( \{ j(x, x') \mapsto x' \cdot x + x \cdot x' \} \)

\[
\begin{align*}
\begin{array}{c}
\text{(x'x)'} = j(x, x') \\
\text{CQ (x'x)'} \geq 0 \iff j(x, x') \geq 0 \\
\text{(x'x)'} \geq 1 \iff j(x, x') \geq 0 \\
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
\text{CE} \quad [x' = x^3][x' := x^3] (x'x) \geq 1' \\
\text{DE} \quad [x' = x^3] (x'x) \geq 1' \\
\text{DI} \quad x'x \geq 1 \to [x' = x^3] x'x \geq 1
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
\text{USR} \quad x^3 \cdot x + x^3 \cdot x \geq 0 \\
\text{USR} \quad x'x \geq 1 \to [x' = x^3] x'x \geq 1
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
\text{US} \quad (x'x) = (x')x + x'(x) \\
\text{US} \quad (x'x) = x'x + x'x
\end{array}
\end{align*}
\]
Example: Differential Invariants Computation

Start with identity differential computation result which proves

Construct differential computation result forward by

Embed differential computation result forward by CT

Construct differential invariant computation result forward accordingly

Resume backward proof with result computed by forward proof right

\[ R(x \cdot x) = (x \cdot x) \cdot x + x \cdot (x \cdot x) \geq 0 \]

\[ x' = x^3 \]

\[ x' = x^3 (x \cdot x \geq 1) \]

\[ x \cdot x \geq 1 \rightarrow [x' = x^3] x \cdot x \geq 1 \]
Start with identity differential computation result

\[
\begin{align*}
\mathbb{R} & \quad (x \cdot x)' = (x \cdot x)' \\
\end{align*}
\]

\[
\begin{align*}
\chi' & \quad \chi' = \chi^3 \\
\end{align*}
\]

\[
\begin{align*}
\text{CT} & \quad \text{Constant Time} \\
\end{align*}
\]

\[
\begin{align*}
\text{CE} & \quad [\chi' = \chi^3][\chi' := \chi^3] (x \cdot x \geq 1)' \\
\end{align*}
\]

\[
\begin{align*}
\text{DE} & \quad [\chi' = \chi^3] (x \cdot x \geq 1)' \\
\end{align*}
\]

\[
\begin{align*}
\text{DI} & \quad x \cdot x \geq 1 \rightarrow [\chi' = \chi^3] x \cdot x \geq 1 \\
\end{align*}
\]

x \cdot x \geq 1 \rightarrow [\chi' = \chi^3] x \cdot x \geq 1
Example: Differential Invariants

1. Start with identity differential computation result which proves

\[ x \cdot x \geq 1 \rightarrow [x' = x^3] x \cdot x \geq 1 \]

\[ 
\begin{array}{l}
\mathbb{R} \\
(x \cdot x)' = (x \cdot x)' \\
. \\
. \\
x' \\
CT \\
\end{array} 
\]

\[ [x' = x^3][x' := x^3](x \cdot x \geq 1)' \]

\[ 
\begin{array}{l}
CE \\
DE \\
DI \\
\end{array} 
\]

\[ 
\begin{array}{l}
[x' = x^3](x \cdot x \geq 1)' \\
\end{array} 
\]
Example: Differential Invariants

1. Start with identity differential computation result which proves
2. Construct differential computation result forward by \( \cdot' \)

\[
\begin{align*}
\mathbb{R} & \quad (x \cdot x)' = (x \cdot x)' \\
\cdot' & \quad (x \cdot x)' = (x)' \cdot x + x \cdot (x)' \\
\cdot & \quad x' \\
\text{CT} & \quad \text{[CE]} \quad [x' = x^3][x' := x^3](x \cdot x \geq 1)' \\
\text{CE} & \quad [x' = x^3](x \cdot x \geq 1)' \\
\text{DE} & \quad x \cdot x \geq 1 \rightarrow [x' = x^3]x \cdot x \geq 1
\end{align*}
\]
Start with identity differential computation result which proves

Construct differential computation result forward by $\cdot' x'$
1. Start with identity differential computation result which proves
2. Construct differential computation result forward by $\dot{x}'$
3. Embed differential computation result forward by CT

\[
\begin{align*}
\mathbb{R} & \quad (x \cdot x)' = (x \cdot x)' \\
\dot{x}' & \quad (x \cdot x)' = (x)' \cdot x + x \cdot (x)' \\
x' & \quad (x \cdot x)' = x' \cdot x + x \cdot x' \\
\mathrm{CT} & \quad (x \cdot x)' \geq 0 \iff x' \cdot x + x \cdot x' \geq 0
\end{align*}
\]

CE

\[
[x' = x^3][x' := x^3](x \cdot x \geq 1)'
\]

DE

\[
[x' = x^3](x \cdot x \geq 1)'
\]

DI

\[
x \cdot x \geq 1 \rightarrow [x' = x^3]x \cdot x \geq 1
\]
1. Start with identity differential computation result which proves
2. Construct differential computation result forward by $x'$
3. Embed differential computation result forward by $\text{CT}$
4. Construct differential invariant computation result forward accordingly

\[
\begin{align*}
\mathbb{R} & \quad \star \\
(x \cdot x)' & = (x \cdot x)' \\
\cdot' & \quad (x \cdot x)' = (x)' \cdot x + x \cdot (x)' \\
\cdot' & \quad (x \cdot x)' = x' \cdot x + x \cdot x' \\
\text{CT} & \quad (x \cdot x)' \geq 0 \iff x' \cdot x + x \cdot x' \geq 0 \\
\text{CE} & \quad [x' = x^3][x' := x^3](x \cdot x \geq 1)' \\
\text{DE} & \quad [x' = x^3](x \cdot x \geq 1)' \\
\text{DI} & \quad x \cdot x \geq 1 \rightarrow [x' = x^3]x \cdot x \geq 1
\end{align*}
\]
Example: Differential Invariants

1. Start with identity differential computation result which proves
2. Construct differential computation result forward by \( \cdot' x' \)
3. Embed differential computation result forward by CT
4. Construct differential invariant computation result forward accordingly
5. Resume backward proof with result computed by forward proof right

\[
\begin{align*}
\mathbb{R} & \quad (x \cdot x)' = (x \cdot x)' \\
\cdot' & \quad (x \cdot x)' = (x)' \cdot x + x \cdot (x)' \\
x' & \quad (x \cdot x)' = x' \cdot x + x \cdot x'
\end{align*}
\]

\[
\begin{align*}
\text{CE} & \quad [x' = x^3][x' := x^3]x' \cdot x + x \cdot x' \geq 0 \\
\text{DE} & \quad [x' = x^3][x' := x^3](x \cdot x \geq 1)' \\
\text{DI} & \quad x \cdot x \geq 1 \rightarrow [x' = x^3]x \cdot x \geq 1
\end{align*}
\]
Example: Differential Invariants

1. Start with identity differential computation result which proves
2. Construct differential computation result forward by \( \cdot' \ x' \)
3. Embed differential computation result forward by CT
4. Construct differential invariant computation result forward accordingly
5. Resume backward proof with result computed by forward proof right

\[
\begin{align*}
\mathbb{R} & : \ (x\cdot x)' = (x\cdot x)' \\
\cdot' & : \ (x\cdot x)' = (x)'\cdot x + x\cdot (x)' \\
x' & : \ (x\cdot x)' = x'\cdot x + x\cdot x'
\end{align*}
\]

\[
\begin{align*}
G_{x' := x^3}[x' \cdot x + x\cdot x' \geq 0] & \quad \text{CT} \quad (x\cdot x)' \geq 0 \iff x'\cdot x + x\cdot x' \geq 0 \\
G_{x' := x^3} x' \cdot x + x\cdot x' \geq 0 & \quad \text{CT} \quad (x\cdot x \geq 1)' \iff x'\cdot x + x\cdot x' \geq 0 \\
[\cdot' :=] & \quad \text{CE} \quad [x' := x^3] (x\cdot x \geq 1)' \\
& \quad \text{DE} \quad [x' := x^3] (x\cdot x \geq 1)' \\
& \quad \text{DI} \quad x\cdot x \geq 1 \rightarrow [x' := x^3] x\cdot x \geq 1
\end{align*}
\]
Example: Differential Invariants

1. Start with identity differential computation result which proves
2. Construct differential computation result forward by $x'$
3. Embed differential computation result forward by CT
4. Construct differential invariant computation result forward accordingly
5. Resume backward proof with result computed by forward proof right

\[
\begin{align*}
\mathbb{R} & \quad \frac{x^3 \cdot x + x \cdot x^3 \geq 0}{\left[ x' := x^3 \right] x' \cdot x + x \cdot x' \geq 0} \\
\mathbb{G} & \quad \frac{\left[ x' = x^3 \right] x' \cdot x + x \cdot x' \geq 0}{\left[ x' = x^3 \right] \left[ x' := x^3 \right] x' \cdot x + x \cdot x' \geq 0} \\
\text{CE} & \quad \frac{\left[ x' = x^3 \right] \left( x \cdot x \geq 1 \right)'}{\left[ x' = x^3 \right] \left( x \cdot x \geq 1 \right)'}
\end{align*}
\]

\[
\begin{align*}
\mathbb{R} & \quad \frac{(x \cdot x)' = (x \cdot x)'}{x'} & \frac{(x \cdot x)' = (x)\cdot x + x \cdot (x)'}{\left[ x' = x^3 \right] x' \cdot x + x \cdot x' \geq 0} \\
\text{CT} & \quad \frac{(x \cdot x)' \geq 0 \iff x' \cdot x + x \cdot x' \geq 0}{\left( x \cdot x \geq 1 \right)' \iff x' \cdot x + x \cdot x' \geq 0} \\
\text{DE} & \quad \frac{\left[ x' = x^3 \right] \left( x \cdot x \geq 1 \right)'}{\left[ x' = x^3 \right] \left( x \cdot x \geq 1 \right)'} \\
\text{DI} & \quad x \cdot x \geq 1 \rightarrow \left[ x' = x^3 \right] x \cdot x \geq 1
\end{align*}
\]
Example: Differential Invariants

1. Start with identity differential computation result which proves
2. Construct differential computation result forward by $\dot{x}'$
3. Embed differential computation result forward by CT
4. Construct differential invariant computation result forward accordingly
5. Resume backward proof with result computed by forward proof right

\[
\begin{align*}
\mathbb{R} & \quad \ast \quad x^3 \cdot x + x \cdot x^3 \geq 0 \\
\mathbb{G} & \quad [x' := x^3] x' \cdot x + x \cdot x' \geq 0 \\
\mathbb{C} & \quad [x' := x^3] x' \cdot x + x \cdot x' \geq 0 \\
\mathbb{D} & \quad [x = x^3] (x \cdot x \geq 1)' \\
\mathbb{E} & \quad [x' = x^3] (x \cdot x \geq 1)' \\
\mathbb{F} & \quad x \cdot x \geq 1 \rightarrow [x' = x^3] x \cdot x \geq 1
\end{align*}
\]
Uniform Substitution

Theorem (Soundness)

\[ \text{US} \quad \frac{\phi}{\sigma(\phi)} \]

provided \( FV(\sigma|_{\Sigma(\theta)}) \cap BV(\otimes(\cdot)) = \emptyset \) for each operation \( \otimes(\theta) \) in \( \phi \)

i.e. bound variables \( U = BV(\otimes(\cdot)) \) of operator \( \otimes \)
are not free in the substitution on its argument \( \theta \) \text{ (U-admissible)}

Uniform substitution \( \sigma \) replaces all occurrences of \( p(\theta) \) for any \( \theta \) by \( \psi(\theta) \)
function \( f(\theta) \) for any \( \theta \) by \( \eta(\theta) \)
quantifier \( C(\phi) \) for any \( \phi \) by \( \psi(\theta) \)
program const. \( a \) by \( \alpha \)

\[ \text{US} \quad [a \cup b]p(\bar{x}) \leftrightarrow [a]p(\bar{x}) \land [b]p(\bar{x}) \]

\[ [x := x + 1 \cup x' = 1]x \geq 0 \leftrightarrow [x := x + 1]x \geq 0 \land [x' = 1]x \geq 0 \]
Uniform Substitution

Theorem (Soundness)

replace all occurrences of \( p(\cdot) \)

\[
\frac{\phi}{\sigma(\phi)}
\]

provided \( \text{FV}(\sigma|_{\Sigma(\theta)}) \cap \text{BV}(\otimes(\cdot)) = \emptyset \) for each operation \( \otimes(\theta) \) in \( \phi \)

i.e. bound variables \( U = \text{BV}(\otimes(\cdot)) \) of operator \( \otimes \)

are not free in the substitution on its argument \( \theta \) \hspace{1cm} (U-admissible)

Uniform substitution \( \sigma \) replaces all occurrences of \( p(\theta) \) for any \( \theta \) by \( \psi(\theta) \)

function \( f(\theta) \) for any \( \theta \) by \( \eta(\theta) \)

quantifier \( C(\phi) \) for any \( \phi \) by \( \psi(\theta) \)

program const. \( a \) by \( \alpha \)

\[
\text{US} \quad [a \cup b]p(\vec{x}) \leftrightarrow [a]p(\vec{x}) \land [b]p(\vec{x})
\]

\[
[x := x + 1 \cup x' = 1]x \geq 0 \leftrightarrow [x := x + 1]x \geq 0 \land [x' = 1]x \geq 0
\]
Lemma (Bound effect lemma) (Only $BV(\cdot)$ change)

If $(u, w) \in \llbracket \alpha \rrbracket I$, then $u = w$ on $BV(\alpha)^C$.

Lemma (Coincidence lemma) (Only $FV(\cdot)$ determine truth)

If $u = \tilde{u}$ on $FV(\theta)$ and $I = J$ on $\Sigma(\theta)$, then $\llbracket \theta \rrbracket lu = \llbracket \theta \rrbracket J\tilde{u}$

$u \in \llbracket \phi \rrbracket I$ iff $\tilde{u} \in \llbracket \phi \rrbracket J$
Lemma (Bound effect lemma)  
If \((u, w) \in \llbracket \alpha \rrbracket I\), then \(u = w\) on \(BV(\alpha)^C\).

Lemma (Coincidence lemma)  
If \(u = \tilde{u}\) on \(FV(\theta)\) and \(I = J\) on \(\Sigma(\theta)\), then \(\llbracket \theta \rrbracket Iu = \llbracket \theta \rrbracket Ju\).  
\(u \in \llbracket \phi \rrbracket I\) iff \(\tilde{u} \in \llbracket \phi \rrbracket J\).
Differential Dynamic Logic $d\mathcal{L}$: Static Semantics

- $\text{FV}((\theta)')$
- $\text{FV}(p(\theta_1, \ldots, \theta_k))$
- $\text{FV}(C(\phi))$
- $\text{FV}(\phi \land \psi)$
- $\text{FV}(\forall x \phi) = \text{FV}(\exists x \phi)$
- $\text{FV}([\alpha]\phi) = \text{FV}(\langle\alpha\rangle\phi)$
- $\text{FV}(a)$
- $\text{FV}(x := \theta) = \text{FV}(x' := \theta)$
- $\text{FV}(?Q)$
- $\text{FV}(x' = f(x) \& Q)$
- $\text{FV}(\alpha \cup \beta)$
- $\text{FV}(\alpha ; \beta)$
- $\text{FV}(\alpha^*)$
Differential Dynamic Logic $\mathcal{DL}$: Static Semantics

\[
\begin{align*}
\text{FV}((\theta)') &= \text{FV}(\theta) \\
\text{FV}(p(\theta_1, \ldots, \theta_k)) &= \text{FV}(\theta_1) \cup \cdots \cup \text{FV}(\theta_k) \\
\text{FV}(C(\phi)) &= \mathcal{V} \cup \mathcal{V}' \\
\text{FV}(\phi \land \psi) &= \text{FV}(\phi) \cup \text{FV}(\psi) \\
\text{FV}(\forall x \, \phi) &= \text{FV}(\exists x \, \phi) = \text{FV}(\phi) \setminus \{x\} \\
\text{FV}([\alpha] \phi) &= \text{FV}(\langle \alpha \rangle \phi) = \text{FV}(\alpha) \cup (\text{FV}(\phi) \setminus \text{BV}(\alpha))
\end{align*}
\]

\[
\begin{align*}
\text{FV}(a) &= \mathcal{V} \cup \mathcal{V}' \\
\text{FV}(x := \theta) &= \text{FV}(x' := \theta) = \text{FV}(\theta) \\
\text{FV}(?Q) &= \text{FV}(Q) \\
\text{FV}(x' = f(x) \& Q) &= \{x\} \cup \text{FV}(f(x)) \cup \text{FV}(Q) \\
\text{FV}(\alpha \cup \beta) &= \text{FV}(\alpha) \cup \text{FV}(\beta) \\
\text{FV}(\alpha; \beta) &= \text{FV}(\alpha) \cup (\text{FV}(\beta) \setminus \text{BV}(\alpha)) \\
\text{FV}(\alpha^*) &= \text{FV}(\alpha)
\end{align*}
\]

for program const. $a$
\[
\text{FV}((\theta)') = \text{FV}(\theta) \cup \text{FV}(\theta)'
\]

\[
\text{FV}(p(\theta_1, \ldots, \theta_k)) = \text{FV}(\theta_1) \cup \cdots \cup \text{FV}(\theta_k)
\]

\[
\text{FV}(C(\phi)) = \mathcal{V} \cup \mathcal{V}'
\]

\[
\text{FV}(\phi \land \psi) = \text{FV}(\phi) \cup \text{FV}(\psi)
\]

\[
\text{FV}(\forall x \phi) = \text{FV}(\exists x \phi) = \text{FV}(\phi) \setminus \{x\}
\]

\[
\text{FV}([\alpha] \phi) = \text{FV}(<\alpha> \phi) = \text{FV}(\alpha) \cup (\text{FV}(\phi) \setminus \text{MBV}(\alpha))
\]

\[
\text{FV}(a) = \mathcal{V} \cup \mathcal{V}'
\]

\[
\text{FV}(x := \theta) = \text{FV}(x' := \theta) = \text{FV}(\theta)
\]

\[
\text{FV}(?Q) = \text{FV}(Q)
\]

\[
\text{FV}(x' = f(x) \& Q) = \{x\} \cup \text{FV}(f(x)) \cup \text{FV}(Q)
\]

\[
\text{FV}(\alpha \cup \beta) = \text{FV}(\alpha) \cup \text{FV}(\beta)
\]

\[
\text{FV}(\alpha; \beta) = \text{FV}(\alpha) \cup (\text{FV}(\beta) \setminus \text{MBV}(\alpha))
\]

\[
\text{FV}(\alpha^*) = \text{FV}(\alpha)
\]
\[ BV(\theta \geq \eta) = BV(p(\theta_1, \ldots, \theta_k)) \]
\[ BV(C(\phi)) \]
\[ BV(\phi \land \psi) \]
\[ BV(\forall x \phi) = BV(\exists x \phi) \]
\[ BV([\alpha]\phi) = BV(\langle \alpha \rangle \phi) \]
\[ BV(a) \]
\[ BV(x := \theta) \]
\[ BV(x' := \theta) \]
\[ BV(?Q) \]
\[ BV(x' = f(x) \& Q) \]
\[ BV(\alpha \cup \beta) = BV(\alpha; \beta) \]
\[ BV(\alpha^*) \]
Differential Dynamic Logic $\mathcal{DL}$: Static Semantics

\[ \text{BV}(\theta \geq \eta) = \text{BV}(p(\theta_1, \ldots, \theta_k)) = \emptyset \]
\[ \text{BV}(C(\phi)) = \mathcal{V} \cup \mathcal{V}' \]
\[ \text{BV}(\phi \land \psi) = \text{BV}(\phi) \cup \text{BV}(\psi) \]
\[ \text{BV}(\forall x \phi) = \text{BV}(\exists x \phi) = \{x\} \cup \text{BV}(\phi) \]
\[ \text{BV}([\alpha]\phi) = \text{BV}([\alpha]\phi) = \text{BV}(\alpha) \cup \text{BV}(\phi) \]

\[ \text{BV}(a) = \mathcal{V} \cup \mathcal{V}' \quad \text{for program constant } a \]
\[ \text{BV}(x := \theta) = \{x\} \]
\[ \text{BV}(x' := \theta) = \{x'\} \]
\[ \text{BV}(?Q) = \emptyset \]
\[ \text{BV}(x' = f(x) \& Q) = \{x, x'\} \]
\[ \text{BV}(\alpha \cup \beta) = \text{BV}(\alpha; \beta) = \text{BV}(\alpha) \cup \text{BV}(\beta) \]
\[ \text{BV}(\alpha^*) = \text{BV}(\alpha) \]
\[ \text{MBV}(a) \]
\[ \text{MBV}(\alpha) \]
\[ \text{MBV}(\alpha \cup \beta) \]
\[ \text{MBV}(\alpha; \beta) \]
\[ \text{MBV}(\alpha^*) \]
Differential Dynamic Logic $\mathcal{DL}$: Static Semantics

\begin{align*}
\text{MBV}(a) &= \emptyset & \text{for program constant } a \\
\text{MBV}(\alpha) &= \text{BV}(\alpha) & \text{for other atomic HPs } \alpha \\
\text{MBV}(\alpha \cup \beta) &= \text{MBV}(\alpha) \cap \text{MBV}(\beta) \\
\text{MBV}(\alpha; \beta) &= \text{MBV}(\alpha) \cup \text{MBV}(\beta) \\
\text{MBV}(\alpha^*) &= \emptyset
\end{align*}
Lemma (Bound effect lemma) (Only $BV(\cdot)$ change)

If $(u, w) \in [\alpha]I$, then $u = w$ on $BV(\alpha)^C$.

Lemma (Coincidence lemma) (Only $FV(\cdot)$ determine truth)

If $u = \tilde{u}$ on $FV(\theta)$ and $I = J$ on $\Sigma(\theta)$, then $[\theta]lu = [\theta]l\tilde{u}$,

$u \in [\phi]I$ iff $\tilde{u} \in [\phi]J$.
Outline

1. CPS are Multi-Dynamical Systems
2. Uniform Substitution Calculus for Differential Dynamic Logic
   - Uniform Substitution Calculus
   - Axiom vs. Axiom Schema
   - Uniform Substitutions
   - Uniform Substitution Lemmas
   - Differential Axioms
   - Differential Invariants
   - Examples
3. Differential-form Differential Dynamic Logic
   - Syntax
   - Semantics
   - Differential Substitution Lemmas
   - Contextual Congruences
   - Parametric Computational Proofs
   - Static Semantics
4. Summary
Uniform Substitution for Differential Dynamic Logic

- Multi-dynamical systems
- Differential forms
- $\leadsto$ local axioms of ODEs
- Uniform substitution
  $\leadsto$ modular generic axioms (not schemata)
- Modular: Logic $\parallel$ Prover
- Straightforward to implement
- Tactics regain efficiency
- Fast contextual equivalence

**dL = DL + HP**
Q: How to build a prover with a small soundness-critical core?
A: Uniform substitution [Church]

Q: How to enable flexible yet sound reasoning?
A: Axioms with local meaning [Philosophy, Algebraic Geometry]

Q: What’s the local meaning of a differential equation?
A: Differential forms [Differential Geometry]

Q: How to do hybrid systems proving?
A: Uniform substitution calculus for differential dynamic logic

Q: What’s the impact of uniform substitution on a prover core?
A: 65,989 \heartsuit 1,677 LOC (2.5%) [KeYmaera X]
Logical Foundations of Cyber-Physical Systems

Logic
- Theorem Proving
- Proofs

Algorithms
- Proof Theory
- Modal Logic
- Model Checking

Analysis
- Lie Algebra
- Algebra
- Differential Algebra

Algebra
- Differential Algebra
- Carathéodory Solutions
- Viscosity PDE Solutions
- Dynamical Systems

Numerics
- Numerical Integration
- Weierstraß Approximation
- Error Analysis

Stochastics
- Doob’s Supermartingales
- Dynkin’s Infinitesimal Generators
- Stochastic Differential Equations

Fixpoints & Lattices
- Decision Procedures
- Closure Ordinals
- Ordinals

Proof Search Procedures
- Theorem Proving
- Proof Theory

Computer Algebra
- Algebraic Geometry
- Algebra

Computer Science
- Decision Procedures
- Fixpoints & Lattices
- Proof Search Procedures

Mathematics
- Differential Equations
- Analysis
- Stochastics
- Numerics
- Proof Theory
- Modal Logic
- Model Checking

Andre Platzer (CMU)

FCPS / 22: Axioms & Uniform Substitutions
<table>
<thead>
<tr>
<th>Approximate LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>KeYmaera X</strong>  1677</td>
</tr>
<tr>
<td><strong>KeYmaera</strong>    65 989</td>
</tr>
<tr>
<td><strong>KeY</strong>         51 328</td>
</tr>
<tr>
<td><strong>HOL Light</strong>   396</td>
</tr>
<tr>
<td><strong>Isabelle/Pure</strong> 8 113</td>
</tr>
<tr>
<td><strong>Nuprl</strong>       15 000 + 50 000</td>
</tr>
<tr>
<td><strong>Coq</strong>         20 000</td>
</tr>
<tr>
<td><strong>HSolver</strong>     20 000</td>
</tr>
<tr>
<td><strong>Flow</strong>        25 000</td>
</tr>
<tr>
<td><strong>PHAVer</strong>      30 000</td>
</tr>
<tr>
<td><strong>dReal</strong>       50 000 + millions</td>
</tr>
<tr>
<td><strong>SpaceEx</strong>     100 000</td>
</tr>
<tr>
<td><strong>HyCreate2</strong>   6 081 + user model analysis</td>
</tr>
</tbody>
</table>

Disclaimer: These self-reported estimates of the soundness-critical lines of code + rules are to be taken with a grain of salt. Different languages, capabilities, styles...
André Platzer.
A uniform substitution calculus for differential dynamic logic.
doi:10.1007/978-3-319-21401-6_32.

André Platzer.
A uniform substitution calculus for differential dynamic logic.

André Platzer.
A complete uniform substitution calculus for differential dynamic logic.

André Platzer.
Logics of dynamical systems.
In LICS [15], pages 13–24.
André Platzer.
Differential dynamic logic for hybrid systems.

André Platzer.
Differential game logic.

André Platzer.
The complete proof theory of hybrid systems.
In LICS [15], pages 541–550.
doi:10.1109/LICS.2012.64.

André Platzer.
A complete axiomatization of quantified differential dynamic logic for distributed hybrid systems.
André Platzer.

André Platzer.

André Platzer and Edmund M. Clarke.
Computing differential invariants of hybrid systems as fixedpoints.  
Special issue for selected papers from CAV’08.  

**André Platzer.**  
The structure of differential invariants and differential cut elimination.  

**André Platzer.**  
A differential operator approach to equational differential invariants.  
doi:10.1007/978-3-642-32347-8_3.

Differential Dynamic Logic: Axioms

\[
\begin{align*}
\mathsf{[=} & \mathsf{]} \quad \mathsf{[x := f]} p(x) \leftrightarrow p(f) \\
\mathsf{[?]} & \quad \mathsf{[?q]} p \leftrightarrow (q \rightarrow p) \\
\mathsf{[\cup]} & \quad \mathsf{[a \cup b]} p(\bar{x}) \leftrightarrow [a]p(\bar{x}) \land [b]p(\bar{x}) \\
\mathsf{[;]} & \quad \mathsf{[a; b]} p(\bar{x}) \leftrightarrow [a][b]p(\bar{x}) \\
\mathsf{[*]} & \quad \mathsf{[a^*]} p(\bar{x}) \leftrightarrow p(\bar{x}) \land [a][a^*]p(\bar{x}) \\
\mathsf{K} & \quad \mathsf{[a]}(p(\bar{x}) \rightarrow q(\bar{x})) \rightarrow ([a]p(\bar{x}) \rightarrow [a]q(\bar{x})) \\
\mathsf{I} & \quad \mathsf{[a^*]}(p(\bar{x}) \rightarrow [a]p(\bar{x})) \rightarrow (p(\bar{x}) \rightarrow [a^*]p(\bar{x})) \\
\mathsf{V} & \quad p \rightarrow [a]p
\end{align*}
\]
Differential Dynamic Logic: Axioms

\[
\begin{align*}
G & : \frac{p(\bar{x})}{[a] p(\bar{x})} \\
\forall & : \frac{p(x)}{\forall x \, p(x)} \\
MP & : \frac{p \rightarrow q \quad p}{q} \\
CT & : \frac{f(\bar{x}) = g(\bar{x})}{c(f(\bar{x})) = c(g(\bar{x}))} \\
CQ & : \frac{f(\bar{x}) = g(\bar{x})}{p(f(\bar{x})) \iff p(g(\bar{x}))} \\
CE & : \frac{p(\bar{x}) \iff q(\bar{x})}{C(p(\bar{x})) \iff C(q(\bar{x}))}
\end{align*}
\]
Differential Equation Axioms & Differential Axioms

\( \text{DW} \ [x' = f(x) \& q(x)] q(x) \)

\( \text{DC} \ ( [x' = f(x) \& q(x)] p(x) \iff [x' = f(x) \& q(x) \land r(x)] p(x) ) \)

\( \text{DE} \ [x' = f(x) \& q(x)] p(x, x') \iff [x' = f(x) \& q(x)][x' := f(x)] p(x, x') \)

\( \text{DI} \ [x' = f(x) \& q(x)] p(x) \iff (q(x) \to p(x) \land [x' = f(x) \& q(x)] (p(x))') \)

\( \text{DG} \ [x' = f(x) \& q(x)] p(x) \iff \exists y [x' = f(x), y' = a(x)y + b(x) \& q(x)] p(x) \)

\( \text{DS} \ [x' = f \& q(x)] p(x) \iff \forall t \geq 0 ((\forall 0 \leq s \leq t q(x+fs)) \to [x := x+ft] p(x)) \)

\( \text{[':=]} \ [x' := f] p(x') \iff p(f) \)

\[ ' (f(x) + g(x)) = (f(x))' + (g(x))' \]

\[ ' (f(x) \cdot g(x)) = (f(x))' \cdot g(x) + f(x) \cdot (g(x))' \]

\[ o' [y := g(x)][y' := 1]((f(g(x)))' = (f(y))' \cdot (g(x))') \]
Axiom (Differential Weakening) (CADE’15)

\[ \text{DW } [x' = f(x) \& q(x)] \neg q(x) \]

Differential equations cannot leave their evolution domains. Implies:

\[ [x' = f(x) \& q(x)] p(x) \leftrightarrow [x' = f(x) \& q(x)] (q(x) \rightarrow p(x)) \]
Axiom (Differential Cut) (CADE’15)

\[
\text{DC} \quad ([x' = f(x) \& q(x)] p(x) \leftrightarrow [x' = f(x) \& q(x) \land r(x)] p(x)) \\
\quad \leftarrow [x' = f(x) \& q(x)] r(x)
\]

DC is a cut for differential equations.
DC is a differential modal modus ponens K.
Can’t leave \( r(x) \), then might as well restrict state space to \( r(x) \).
Axiom (Differential Cut) (CADE’15)

\[
\begin{align*}
\text{DC} & \quad ([x' = f(x) & q(x)] p(x) \leftrightarrow [x' = f(x) & q(x) \land r(x)] p(x)) \\
& \quad \leftarrow [x' = f(x) & q(x)] r(x)
\end{align*}
\]

DC is a cut for differential equations.
DC is a differential modal modus ponens K.
Can’t leave \( r(x) \), then might as well restrict state space to \( r(x) \).
Axiom (Differential Cut) \hspace{1cm} (CADE’15)

\[
\text{DC} \quad \begin{array}{c}
([x' = f(x) \& q(x)]p(x) \leftrightarrow [x' = f(x) \& q(x) \land r(x)]p(x)) \\
\leftarrow [x' = f(x) \& q(x)]r(x)
\end{array}
\]

DC is a cut for differential equations.
DC is a differential modal modus ponens K.
Can’t leave \( r(x) \), then might as well restrict state space to \( r(x) \).
Differential Equation Axioms

Axiom (Differential Cut) (CADE’15)

\[
\text{DC} \quad ([x' = f(x) \& q(x)]p(x) \leftrightarrow [x' = f(x) \& q(x) \land r(x)]p(x)) \\
\quad \leftarrow [x' = f(x) \& q(x)]r(x)
\]

DC is a cut for differential equations.
DC is a differential modal modus ponens K.
Can’t leave \(r(x)\), then might as well restrict state space to \(r(x)\).
Axiom (Differential Cut) (CADE’15)

\[ [x' = f(x) & q(x)]p(x) \leftrightarrow [x' = f(x) & q(x) \land r(x)]p(x) \]
\[ \leftarrow [x' = f(x) & q(x)]r(x) \]

DC is a cut for differential equations.
DC is a differential modal modus ponens K.
Can’t leave \( r(x) \), then might as well restrict state space to \( r(x) \).
Differential Equation Axioms

Axiom (Differential Cut) (CADE’15)

DC

\[
\begin{align*}
[x' = f(x) & q(x)]p(x) & \iff [x' = f(x) & q(x) \land r(x)]p(x) \\
& \iff [x' = f(x) & q(x)]r(x)
\end{align*}
\]

DC is a cut for differential equations.
DC is a differential modal modus ponens K.
Can’t leave \( r(x) \), then might as well restrict state space to \( r(x) \).
Differential Equation Axioms

Axiom (Differential Cut) (CADE’15)

\[
\begin{align*}
DC & \quad ([x' = f(x) \land q(x)] p(x) \leftrightarrow [x' = f(x) \land q(x) \land r(x)] p(x)) \\
& \quad \leftarrow [x' = f(x) \land q(x)] r(x)
\end{align*}
\]

DC is a cut for differential equations.
DC is a differential modal modus ponens K.
Can’t leave \( r(x) \), then might as well restrict state space to \( r(x) \).
Axiom (Differential Cut) (CADE’15)

\[
\begin{align*}
DC & \quad \left( [x' = f(x) \& q(x)]p(x) \leftrightarrow [x' = f(x) \& q(x) \& r(x)]p(x) \right) \\
& \quad \leftarrow [x' = f(x) \& q(x)]r(x)
\end{align*}
\]

DC is a cut for differential equations.
DC is a differential modal modus ponens K.
Can’t leave \( r(x) \), then might as well restrict state space to \( r(x) \).
Axiom (Differential Cut) (CADE'15)

\[
\text{DC} \quad ([x' = f(x) \& q(x)] p(x) \leftrightarrow [x' = f(x) \& q(x) \& r(x)] p(x))
\]
\[
\leftrightarrow [x' = f(x) \& q(x)] r(x)
\]

DC is a cut for differential equations.
DC is a differential modal modus ponens K.
Can’t leave \( r(x) \), then might as well restrict state space to \( r(x) \).
Axiom (Differential Invariant) (CADE’15)

\[ \text{DI } [x' = f(x) \& q(x)]p(x) \leftrightarrow (q(x) \rightarrow p(x) \land [x' = f(x) \& q(x)](p(x)))' \]

Differential invariant: \( p(x) \) true now and its differential \((p(x))'\) true always

What’s the differential of a formula???
What’s the meaning of a differential term . . . in a state???
Axiom (Differential Effect) \(\text{(CADE'15)}\)

\[
\text{DE} \quad [x' = f(x) \& q(x)]p(x, x') \iff [x' = f(x) \& q(x)][x' := f(x)]p(x, x')
\]

Effect of differential equation on differential symbol \(x'\)

\([x' := f(x)]\) instantly mimics continuous effect \([x' = f(x)]\) on \(x'\)

\([x' := f(x)]\) selects vector field \(x' = f(x)\) for subsequent differentials
Differential Equation Axioms

Axiom (Differential Ghost) (CADE’15)

\[ DG \quad [x' = f(x) & q(x)]p(x) \leftrightarrow \exists y \quad [x' = f(x), y' = a(x)y + b(x) & q(x)]p(x) \]

Differential ghost/auxiliaries: extra differential equations that exist
Can cause new invariants
“Dark matter” counterweight to balance conserved quantities
Axiom (Differential Solution) \(\text{(CADE'15)}\)

\[
\text{DS} \quad \left[ x' = f & q(x) \right] p(x) \iff \forall t \geq 0 \left( \left( \forall 0 \leq s \leq t q(x + fs) \right) \rightarrow [x := x + ft] p(x) \right)
\]

### Definition (Term semantics)

\[
\llbracket (\theta)' \rrbracket_l u = \sum_x u(x') \frac{\partial \llbracket \theta \rrbracket}{\partial x}(u) = \sum_x u(x') \frac{\partial \llbracket \theta \rrbracket l u^X}{\partial X}
\]

### Definition (\(d\mathcal{L}\) semantics)

\[
\begin{align*}
\llbracket C(\phi) \rrbracket_l &= l(C)(\llbracket \phi \rrbracket_l) \\
\llbracket \alpha \phi \rrbracket_l &= \llbracket \alpha \rrbracket_l \circ \llbracket \phi \rrbracket_l \\
\llbracket [\alpha] \phi \rrbracket_l &= \llbracket \neg \langle \alpha \rangle \neg \phi \rrbracket_l
\end{align*}
\]

### Definition (Program semantics)

\[
\begin{align*}
\llbracket x' = f(x) \& Q \rrbracket_l &= \{(\varphi(0)|_{x'}^c, \varphi(r)) : l, \varphi \models x' = f(x) \land Q\} \\
\llbracket \alpha \cup \beta \rrbracket_l &= \llbracket \alpha \rrbracket_l \cup \llbracket \beta \rrbracket_l \\
\llbracket \alpha ; \beta \rrbracket_l &= \llbracket \alpha \rrbracket_l \circ \llbracket \beta \rrbracket_l \\
\llbracket \alpha^* \rrbracket_l &= (\llbracket \alpha \rrbracket_l)^* = \bigcup_{n \in \mathbb{N}} \llbracket \alpha^n \rrbracket_l
\end{align*}
\]
### Differential-form Differential Dynamic Logic: Semantics

#### Definition (Term semantics) \((\llbracket \cdot \rrbracket : \text{Trm} \rightarrow (S \rightarrow \mathbb{R}))\)

- \(\llbracket x \rrbracket lu = u(x)\) for variable \(x \in V\)
- \(\llbracket x' \rrbracket lu = u(x')\) for differential symbol \(x' \in V'\)
- \(\llbracket f(\theta_1, \ldots, \theta_k) \rrbracket lu = l(f)(\llbracket \theta_1 \rrbracket lu, \ldots, \llbracket \theta_k \rrbracket lu)\) for function symbol \(f\)
- \(\llbracket \theta + \eta \rrbracket lu = \llbracket \theta \rrbracket lu + \llbracket \eta \rrbracket lu\)
- \(\llbracket \theta \cdot \eta \rrbracket lu = \llbracket \theta \rrbracket lu \cdot \llbracket \eta \rrbracket lu\)
- \(\llbracket (\theta)' \rrbracket lu = \sum_x u(x') \frac{\partial \llbracket \theta \rrbracket lu}{\partial x}(u)\) for variable \(x \in V\)
- \(\llbracket (\theta)' \rrbracket lu = \sum_x u(x') \frac{\partial \llbracket \theta \rrbracket lu_X}{\partial X}(u)\) for differential symbol \(x' \in V'\)

#### Definition (dL semantics) \((\llbracket \cdot \rrbracket : \text{Fml} \rightarrow \wp(S))\)

- \(\llbracket C(\phi) \rrbracket l = l(C)(\llbracket \phi \rrbracket l)\)
- \(\llbracket \langle \alpha \rangle \phi \rrbracket l = \llbracket \alpha \rrbracket l \circ \llbracket \phi \rrbracket l\)
- \(\llbracket [\alpha] \phi \rrbracket l = \llbracket \neg \langle \alpha \rangle \neg \phi \rrbracket l\)

#### Definition (Program semantics) \((\llbracket \cdot \rrbracket : \text{HP} \rightarrow \wp(S \times S))\)

- \(\llbracket x' = f(x) \land Q \rrbracket l = \{ (\phi(0)|\{x'\}\setminus Q, \phi(r)) : l, \phi|\frac{\partial \llbracket \theta \rrbracket lu_X}{\partial X}\} \}
- \(\llbracket \alpha \cup \beta \rrbracket l = \llbracket \alpha \rrbracket l \cup \llbracket \beta \rrbracket l\)
- \(\llbracket \alpha ; \beta \rrbracket l = \llbracket \alpha \rrbracket l \circ \llbracket \beta \rrbracket l\)
- \(\llbracket \alpha \ast \rrbracket l = (\llbracket \alpha \rrbracket l)^\ast = \bigcup_{n \in \mathbb{N}} \llbracket \alpha_n \rrbracket l\)

---

**Andre Platzer (CMU)**

FCPS / 22: Axioms & Uniform Substitutions
Definition (Term semantics)  
$$([(\theta)'])_{lu} = \sum_{x} u(x') \frac{\partial [\theta]}{\partial x} (u) = \sum_{x} u(x') \frac{\partial [\theta]_{lu}}{\partial x}$$

Definition (dL semantics)  
$$[[\theta \geq \eta]]_{I} = \{ u : [[\theta]]_{lu} \geq [[\eta]]_{lu} \}$$
$$[[p(\theta_1, \ldots, \theta_k)]]_{I} = \{ u : ([[\theta_1]]_{lu}, \ldots, [\theta_k]_{lu}) \in I(p) \}$$
$$[[C(\phi)]]_{I} = l(C)([[\phi]]_{I})$$
$$[[\neg \phi]]_{I} = ([[\phi]]_{I})^C$$
$$[[\phi \land \psi]]_{I} = [[\phi]]_{I} \cap [[\psi]]_{I}$$
$$[[\exists x \phi]]_{I} = \{ u \in S : u'_x \in [[\phi]]_{I} \text{ for some } r \in \mathbb{R} \}$$
$$[[\langle \alpha \rangle \phi]]_{I} = [[\alpha]]_{I} \circ [[\phi]]_{I} = \{ u : w \in [[\phi]]_{I} \text{ for some } w (u, w) \in [[\alpha]]_{I} \}$$
$$[[[\alpha] \phi]]_{I} = [[\neg \langle \alpha \rangle \neg \phi]]_{I} = \{ u : w \in [[\phi]]_{I} \text{ for all } w (u, w) \in [[\alpha]]_{I} \}$$

Definition (Program semantics)  
$$([[]] : HP \rightarrow \wp(S \times S))$$
$$[[x'] = f(x) \& Q]]_{I} = \{ (\phi(0) | \{ x' \in \mathbb{R}, \phi(\bar{r}) \}) : I, \phi | = x' = f(x) \land Q \}$$
Definition (Term semantics) \((\llbracket \cdot \rrbracket : \text{Trm} \to (\mathcal{S} \to \mathbb{R}))\)

\[
\llbracket (\theta)' \rrbracket Iu = \sum_x u(x') \frac{\partial \llbracket \theta \rrbracket I}{\partial x} (u) = \sum_x u(x') \frac{\partial \llbracket \theta \rrbracket Iu^x}{\partial X}
\]

Definition (\(d\mathcal{L}\) semantics) \((\llbracket \cdot \rrbracket : \text{Fml} \to \wp(\mathcal{S}))\)

\[
\llbracket C(\phi) \rrbracket I = I(C)(\llbracket \phi \rrbracket I)
\]
\[
\llbracket \langle \alpha \rangle \phi \rrbracket I = \llbracket \alpha \rrbracket I \circ \llbracket \phi \rrbracket I
\]
\[
\llbracket [\alpha] \phi \rrbracket I = \llbracket \neg \langle \alpha \rangle \neg \phi \rrbracket I
\]

Definition (Program semantics) \((\llbracket \cdot \rrbracket : \text{HP} \to \wp(\mathcal{S} \times \mathcal{S}))\)

\[
\llbracket a \rrbracket I = I(a)
\]
\[
\llbracket x := \theta \rrbracket I = \{(u, w) : w = u \text{ except } \llbracket x \rrbracket Iw = \llbracket \theta \rrbracket Iu\}
\]
\[
\llbracket x' := \theta \rrbracket I = \{(u, w) : w = u \text{ except } \llbracket x' \rrbracket Iw = \llbracket \theta \rrbracket Iu\}
\]
\[
\llbracket ?Q \rrbracket I = \{(u, u) : u \in \llbracket Q \rrbracket I\}
\]
\[
\llbracket x' = f(x) \& Q \rrbracket I = \{(\varphi(0)|_x, \varphi(r)) : I, \varphi \models x' = f(x) \land Q\}
\]
\[
\llbracket \alpha \cup \beta \rrbracket I = \llbracket \alpha \rrbracket I \cup \llbracket \beta \rrbracket I
\]