Units of Measure for Hybrid Programs

15-424 Final Project

Vincent Huang (vincenth@andrew.cmu.edu)

Carnegie Mellon University

May 5, 2016
Hybrid systems, differential dynamic logic, and KeYmaera X

This is a hybrid program, which models a hybrid system

It is a formula in $\mathcal{dL}$

KeYmaera X is a theorem prover for $\mathcal{dL}$
Hybrid systems, differential dynamic logic, and KeYmaera X

\([\alpha]^P\]

- This is a **hybrid program**, which models a **hybrid system**
- It is a **formula** in **dŁ**
- **KeYmaera X** is a theorem prover for **dŁ**
Hybrid systems, differential dynamic logic, and KeYmaera X

\[
[\alpha]P
\]

- This is a hybrid program, which models a hybrid system
- It is a formula in dL
- KeYmaera X is a theorem prover for dL
Verifying a model vs validating it

- **Verification** can be done in the proof calculus of dL (and can be partially automated/fully checked by KeYmaera X)
- **Validation** is checking if a model is actually representative of the system it’s supposed to be modelling
- A difficult problem, *but* there are some things we can do purely syntactically—including this!
Verifying a model vs validating it

- **Verification** can be done in the proof calculus of dŁ (and can be partially automated/fully checked by KeYmaera X)
- **Validation** is checking if a model is actually representative of the system it’s supposed to be modelling

  A difficult problem, *but* there are some things we can do purely syntactically—including this!
Verifying a model vs validating it

- **Verification** can be done in the proof calculus of $d\mathcal{L}$ (and can be partially automated/fully checked by KeYmaera X)
- **Validation** is checking if a model is actually representative of the system it’s supposed to be modelling
- A difficult problem, *but* there are some things we can do purely syntactically—including this!
Example 1: The weirdly-accelerating car

Consider the hybrid program on the right, meant to model a 1D car that has to stop before some point

ODE, instead of $x' = v$, has $x' = x$ (which doesn’t make sense physically)

Serious mistake; model is not a car moving in a straight line

But it passes muster in KeYmaera X 4.2b1!

User might waste plenty of time trying to verify an unprovable model

---

\[
\begin{align*}
\text{Problem.} & \quad \left[ \left\{ \; x' = x \; \right\} \right] x \leq S \\
\text{End.} & \quad \text{Figure: An incorrect model of a 1D car}
\end{align*}
\]
Example 1: The weirdly-accelerating car

- Consider the hybrid program on the right, meant to model a 1D car that has to stop before some point
- ODE, instead of $x' = v$, has $x' = x$ (which doesn’t make sense physically)
- Serious mistake; model is not a car moving in a straight line
- But it passes muster in KeYmaera X 4.2b1!
- User might waste plenty of time trying to verify an unprovable model

ProgramVariables.

R x.
R S.
End.

Problem.

\[
\{ x' = x \} x \leq S
\]

End.

Figure: An incorrect model of a 1D car
Example 1: The weirdly-accelerating car

- Consider the hybrid program on the right, meant to model a 1D car that has to stop before some point
- ODE, instead of $x' = v$, has $x' = x$ (which doesn’t make sense physically)
- Serious mistake; model is not a car moving in a straight line
- But it passes muster in KeYmaera X 4.2b1!

User might waste plenty of time trying to verify an unprovable model

```
ProgramVariables.
  R x.
  R S.
End.
Problem.
  [\{ x' = x \}] x <= S
End.
```

**Figure:** An incorrect model of a 1D car
Example 1: The weirdly-accelerating car

Consider the hybrid program on the right, meant to model a 1D car that has to stop before some point.

ODE, instead of \( x' = v \), has \( x' = x \) (which doesn’t make sense physically).

Serious mistake; model is not a car moving in a straight line.

But it passes muster in KeYmaera X 4.2b1!

User might waste plenty of time trying to verify an unprovable model.

\[
\begin{align*}
\text{ProgramVariables.} \\
R &\ x. \\
R &\ S. \\
\text{End.} \\
\text{Problem.} \\
[\{ \ x' = x \ \}]x &\leq S \\
\text{End.} \\
\end{align*}
\]

Figure: An incorrect model of a 1D car.
Example 2: The neat new distance metric

- Lab 3 of 15-424 involved modelling a robot travelling around a circular track and verifying that it would stop before it ran into an obstacle.
- A student submitted a model hinging on the test given below:
  - Subtracts a quantity with dimension L from a quantity with dimension L^2
  - Subtle problem—mostly a safe if unnecessarily conservative overapproximation, except if ox − x < 1? What happens then?

```
?((ox-x)^2 + (oy-y)^2 - v*T - (a*T^2)/2
>= -(v + a*T)^2)/(2*B));
```

**Figure:** An incorrect test
Example 2: The neat new distance metric

- Lab 3 of 15-424 involved modelling a robot travelling around a circular track and verifying that it would stop before it ran into an obstacle.
- A student submitted a model hinging on the test given below:
  - Subtracts a quantity with dimension L from a quantity with dimension $L^2$.
  - Subtle problem—mostly a safe if unnecessarily conservative overapproximation, except if $ox - x < 1$? What happens then?

$$?((ox-x)^2 + (oy-y)^2 - v*T - (a*T^2)/2 \\
\geq -((v + a*T)^2)/(2*B));$$

**Figure:** An incorrect test
Example 2: The neat new distance metric

- Lab 3 of 15-424 involved modelling a robot travelling around a circular track and verifying that it would stop before it ran into an obstacle.
- A student submitted a model hinging on the test given below:
- Subtracts a quantity with dimension L from a quantity with dimension L^2
- Subtle problem—mostly a safe if unnecessarily conservative overapproximation, except if ox - x < 1? What happens then?

\[ \frac{(ox-x)^2 + (oy-y)^2 - v*T - (a*T^2)/2}{(v + a*T)^2/(2*B)} \geq 0 \]

**Figure:** An incorrect test
Example 2: The neat new distance metric

- Lab 3 of 15-424 involved modelling a robot travelling around a circular track and verifying that it would stop before it ran into an obstacle.
- A student submitted a model hinging on the test given below:
- Subtracts a quantity with dimension L from a quantity with dimension \( L^2 \)
- Subtle problem—mostly a safe if unnecessarily conservative overapproximation, except if \( ox - x < 1 \)? What happens then?

\[
?\left( (ox-x)^2 + (oy-y)^2 - v*T - (a*T^2)/2 \right) 
\geq -((v + a*T)^2)/(2*B));
\]

**Figure:** An incorrect test
How do we fix this?

- Units of measure!
- Physics people realise that comparing incommensurate quantities doesn’t make sense
- CPS verification is “physics stuff”!
The weirdly-accelerating car revisited

We can annotate the incorrect model of the 1D car with units, and see what happens!

ProgramUnits.
    U m.
End.
ProgramVariables.
    R x : m.
    R S : m.
End.
Problem.
    [{ x' = x }] x <= S
End.

Figure: The incorrect model of the 1D car with unit annotations
The weirdly-accelerating car revisited

ProgramUnits.
  U m.
End.
ProgramVariables.
  R x : m.
  R S : m.
End.
Problem.
  [{ x' = x }]x <= S
End.

Figure: The incorrect model of the 1D car with unit annotations

Unit analysis error
Units do not match in expression x' = x
x' = x
  ^ m*s^(-1)
  ^ m
The neat new distance metric revisited

We isolate the problematic test in an annotated hybrid program, and see what happens.

ProblemUnits.  Problem.
U m.  [?((ox-x)^2 + (oy-y)^2 - v*T
End.  - (a*T^2)/2
ProblemVariables.  >= -((v + a*T)^2)/(2*B));]x=x
R ox : m.  End.
R x : m.
R oy : m.
R y : m.
R a : m/(s*s).
R T : s.
R v : m/s.
R A : m/(s*s).
R B : m/(s*s).
End.
The neat new distance metric revisited

ProgramUnits.
  U m.
End.

ProgramVariables.
  R ox : m.
  R x : m.
  R oy : m.
  R y : m.
  R a : m/(s*s).
  R T : s.
  R v : m/s.
  R A : m/(s*s).
  R B : m/(s*s).
End.

Problem.
  \[ ?((ox-x)^2 + (oy-y)^2 - v*T - (a*T^2)/2 \geq -((v + a*T)^2)/(2*B)); \] x=x
End.

Unit analysis error
unit error in term on LHS of \( \geq \)
  Problematic term is \((ox-x)^2+(oy-y)^2-v*T-a*T^2/2\)
Units for $d\mathcal{L}$ and KeYmaera X

- We developed a monomorphic unit-of-measure type system with a top type for $d\mathcal{L}$
- We built a working implementation in the current version of KeYmaera X
- Our version of KeYmaera X is entirely backward-compatible with the existing version—the presence of $\top$ in the type system means that we can assign any unannotated variables type $\top$ and hence programs written without explicit units still typecheck.
Units for $d\mathcal{L}$ and KeYmaera X

- We developed a monomorphic unit-of-measure type system with a top type for $d\mathcal{L}$
- We built a working implementation in the current version of KeYmaera X
- Our version of KeYmaera X is entirely backward-compatible with the existing version—the presence of $\top$ in the type system means that we can assign any unannotated variables type $\top$ and hence programs written without explicit units still typecheck.
Some rules for checking terms

**Times-T** \( \Gamma \vdash x_1 : \tau_1 \quad \Gamma \vdash x_2 : \tau_2 \)
\[
\Gamma \vdash x_1 \times x_2 : \tau_1 \cdot \tau_2
\]

**Div-T** \( \Gamma \vdash x_1 : \tau_1 \quad \Gamma \vdash x_2 : \tau_2 \)
\[
\Gamma \vdash x_1 \div x_2 : \tau_1 \cdot \tau_2^{-1}
\]

**Figure:** Representative examples of rules for typing \( d\mathcal{L} \) terms
Unit-checking $\mathcal{dL}$

Example (times)

\[
\begin{align*}
\text{Var-T} & \quad \gamma(x) = m \\
\text{Times-T} & \quad \gamma \vdash x : m \\
\text{Var-T} & \quad \gamma(y) = m \\
\text{Var-T} & \quad \gamma \vdash y : m \\
\gamma \vdash x \cdot y : m^2
\end{align*}
\]
Unit-checking dL

Example (divide)

\[
\frac{\Var-T}{\text{Times-T}} \quad \frac{\gamma(x) = m}{\gamma \vdash x : m} \quad \frac{\gamma(t) = s}{\gamma \vdash t : s} \Rightarrow \gamma \vdash x / y : m \cdot s^{-1}
\]
Checking $d\mathcal{L}$ formulas

Rule

\[
\begin{array}{c}
\frac{\gamma \vdash t_1 : \tau \qquad \gamma \vdash t_2 : \tau}{\gamma \vdash t_1 = t_2 : \tau}
\end{array}
\]

Figure: Representative example of rules for validating $d\mathcal{L}$ formulas
Checking dŁ formulas

Example

\[ \vdash x : m \quad \vdash y : m \]
\[ \vdash x = y \quad \text{ok} \]
Checking dL programs

\[ \vdash P_1 \text{ runs} \quad \vdash P_2 \text{ runs} \]
\[ \vdash P_1 ; P_2 \text{ runs} \]

ODE-runs
\[ \vdash x : \tau \quad \vdash t : \tau \cdot s^{-1} \]
\[ \vdash \{ x' = t \} \text{ runs} \]

Figure: Representative examples of rules for validating dL programs
Adding units to KeYmaera X

- We implemented unit of measure types and a unit-checker in KeYmaera X in accordance with the rules given on previous slides.
- Only very minor modifications to the KeYmaera X core (the soundness-critical part of KeYmaera X)!
- Only addition of a new datatype to expressions to support units
- If you trusted KeYmaera X previously, you can still trust it now!
The normally-accelerating car

ProgramUnits.
  U m.
End.
ProgramVariables.
  R x : m.
  R v : m/s.
  R S : m.
End.
Problem.
  [{ x' = v }]x <= S
End.

Figure: A corrected model of the 1D car. Will pass the unit checker!
Future work

- Fruitful avenue of validating hybrid systems models without having to build the real system
- Units of measure lend themselves to other interesting applications within KeYmaera X
  - Constraining invariant/proof search?
  - Improved user interface?
Conclusion

- Unit analysis is easy
- ... for computers!
- Fully automatable, and fully automated!
- Find more bugs today!
Conclusion

- Unit analysis is easy
- ... for computers!
- Fully automatable, and fully automated!
- Find more bugs today!
Conclusion

- Unit analysis is easy
- ... for computers!
- Fully automatable, and fully automated!
- Find more bugs today!
Conclusion

- Unit analysis is easy
- ... for computers!
- Fully automatable, and fully automated!
- Find more bugs today!
Questions?