Definitely Not Flawed or Not Definitely Flawed?

FPL Away Days 2013

Henrik Nilsson

Joint work with John Capper

School of Computer Science

University of Nottingham

How to Design a Type System? (1)

- Commonly, we expect well-typedness to be a proof of some appropriate property of goodness:

  "the program is definitely not flawed"

How to Design a Type System? (1)

- Commonly, we expect well-typedness to be a proof of some appropriate property of goodness:

  "the program is definitely not flawed"

- This underpins notions such as progress and preservation; it would be strange if a flawless program turns out to be flawed once it is run:

  "Well-typed programs do not go wrong

How to Design a Type System? (1)

- Commonly, we expect well-typedness to be a proof of some appropriate property of goodness:

  “the program is definitely not flawed”

- This underpins notions such as progress and preservation; it would be strange if a flawless program turns out to be flawed once it is run:

  “Well-typed programs do not go wrong [and should therefore be accepted]”

How to Design a Type System? (1)

- Commonly, we expect well-typedness to be a proof of some appropriate property of goodness:

  “the program is definitely not flawed”

- This underpins notions such as progress and preservation; it would be strange if a flawless program turns out to be flawed once it is run:

  “Well-typed programs do not go wrong [and should therefore be accepted]”

- Of course, depending, we may have to settle for a quite weak notion of “flawless”.

How to Design a Type System? (2)

But is that really the ***only*** option?

How to Design a Type System? (2)

But is that really the ***only*** option?

- Assume main design goal is to catch as many problems as possible as early as possible.

How to Design a Type System? (2)

But is that really the ***only*** option?

- Assume main design goal is to catch as many problems as possible as early as possible.
- Then what about opting for

  "the program is not definitely flawed"

  if that allows a ***stronger*** notion of "flawed", meaning ***more*** problems can be caught?
- Of course, a not definitely flawed program may be revealed to be flawed; e.g. when run.

How to Design a Type System? (3)

- One may question if we then are entitled to talk about a “type system”; Pierece (2002):

  A type system is a tractable syntactic method for proving the absence of certain [undesirable] program behaviors by classifying phrases according to the kinds of values they compute.

How to Design a Type System? (3)

- One may question if we then are entitled to talk about a “type system”; Pierece (2002):

  A type system is a tractable syntactic method for proving the absence of certain [undesirable] program behaviors by classifying phrases according to the kinds of values they compute.

- What we are considering here is proving the ***presence*** of undesirable program behaviours, and rule out the program if so.

How to Design a Type System? (4)

- Or, in other words, we acknowledge the obvious:

How to Design a Type System? (4)

- Or, in other words, we acknowledge the obvious:

  Ill-typed programs can go wrong

How to Design a Type System? (4)

- Or, in other words, we acknowledge the obvious:

  Ill-typed programs can go wrong (and must therefore be rejected)

How to Design a Type System? (4)

- Or, in other words, we acknowledge the obvious:

  Ill-typed programs can go wrong (and must therefore be rejected)

  but do not insist that well-typed programs do not reveal further errors when run.

How to Design a Type System? (4)

- Or, in other words, we acknowledge the obvious:

  Ill-typed programs can go wrong (and must therefore be rejected)

  but do not insist that well-typed programs do not reveal further errors when run.
- A kind of mixed static and dynamic approach to typing.

How to Design a Type System? (4)

- Or, in other words, we acknowledge the obvious:

  Ill-typed programs can go wrong (and must therefore be rejected)

  but do not insist that well-typed programs do not reveal further errors when run.
- A kind of mixed static and dynamic approach to typing.
- Possible to view as instance of other “mixed approaches”, such as Hybrid Type Checking (Flanagan) or Gradual Typing (Siek and Taha)?

This Talk

- Overview of the equation-system aspect of a type system for Functional Hybrid Modelling (FHM) as an example of opting for “not definitely flawed”.

This Talk

- Overview of the equation-system aspect of a type system for Functional Hybrid Modelling (FHM) as an example of opting for “not definitely flawed”.
- Possible points for discussion:
  - Pros and cons of the approach.
  - What meta-theoretical properties can or should one prove?
  - Other instances of (aspects of) type systems like this?

The FHM Type System

- ***Refinement*** of a standard type system:

The FHM Type System

- ***Refinement*** of a standard type system:
  - Progress and preservation holds for the ***unrefined*** system.

The FHM Type System

- ***Refinement*** of a standard type system:
  - Progress and preservation holds for the ***unrefined*** system.
- The refinements impose ***additional*** constraints related to “well-formedness” of modular systems of equations:

The FHM Type System

- ***Refinement*** of a standard type system:
  - Progress and preservation holds for the ***unrefined*** system.
- The refinements impose ***additional*** constraints related to “well-formedness” of modular systems of equations:
  - The refined system rules out ***strictly more*** programs than the unrefined one.

The FHM Type System

- ***Refinement*** of a standard type system:
  - Progress and preservation holds for the ***unrefined*** system.
- The refinements impose ***additional*** constraints related to “well-formedness” of modular systems of equations:
  - The refined system rules out ***strictly more*** programs than the unrefined one.
  - In particular, definitely ill-formed equation system fragments are rejected.

The FHM Type System

- ***Refinement*** of a standard type system:
  - Progress and preservation holds for the ***unrefined*** system.
- The refinements impose ***additional*** constraints related to “well-formedness” of modular systems of equations:
  - The refined system rules out ***strictly more*** programs than the unrefined one.
  - In particular, definitely ill-formed equation system fragments are rejected.
  - But there is ***no guarantee*** that the resulting system of equations will be well-formed.

The FHM Setting (1)

- FHM is a functional approach to modelling and simulation of (physical) systems that can be described by an ***evolving*** set of differential equations.

The FHM Setting (1)

- FHM is a functional approach to modelling and simulation of (physical) systems that can be described by an ***evolving*** set of differential equations.
- Undirected equations: ***non-causal modelling***. (Differential Algebraic Equations, DAE)

The FHM Setting (1)

- FHM is a functional approach to modelling and simulation of (physical) systems that can be described by an ***evolving*** set of differential equations.
- Undirected equations: ***non-causal modelling***. (Differential Algebraic Equations, DAE)
- Two-level design:
  - ***equation level*** for modelling components
  - ***functional level*** for spatial and temporal composition of components

The FHM Setting (2)

- Equations system fragments are first-class entities at the functional level; viewed as relations on signal, or ***signal relations***.

The FHM Setting (2)

- Equations system fragments are first-class entities at the functional level; viewed as relations on signal, or ***signal relations***.
- Spatial composition: signal relation ***application***; enables modular, hierarchical, system description.

The FHM Setting (2)

- Equations system fragments are first-class entities at the functional level; viewed as relations on signal, or ***signal relations***.
- Spatial composition: signal relation ***application***; enables modular, hierarchical, system description.
- Temporal composition: ***switching*** from one structural configuration into another.

The FHM Setting (3)

$$\begin{aligned}
&resistor :: \texttt{Resistance} \rightarrow \mathsf{SR}\,(\texttt{Pin}, \texttt{Pin})\\
&resistor\ r = \textbf{sigrel}\ (p, n)\ \textbf{where}\\
&\qquad twoPin \diamond (p, n, u)\\
&\qquad r \cdot p.i = u
\end{aligned}$$

The FHM Setting (3)

Parametrised model represented by ***function*** mapping parameters to a model. Note: first class models!

$$
\begin{aligned}
&resistor :: \texttt{Resistance} \rightarrow \textsf{SR}\,(\texttt{Pin}, \texttt{Pin}) \\
&resistor\ r = \textbf{sigrel}\ (p, n)\ \textbf{where} \\
&\qquad twoPin \diamond (p, n, u) \\
&\qquad r \cdot p.i = u
\end{aligned}
$$

The FHM Setting (3)

Parametrised model represented by ***function*** mapping parameters to a model. Note: first class models!

$$resistor :: \texttt{Resistance} \rightarrow \texttt{SR}\,(\texttt{Pin}, \texttt{Pin})$$
$$resistor\ r = \textbf{sigrel}\ (p, n)\ \textbf{where}$$
$$twoPin \diamond (p, n, u)$$
$$r \cdot p.i = u$$

Encapsulated equation system fragment.

The FHM Setting (3)

Parametrised model represented by ***function*** mapping parameters to a model. Note: first class models!

$$resistor :: \texttt{Resistance} \rightarrow \mathsf{SR}\,(\texttt{Pin}, \texttt{Pin})$$

$$\begin{aligned} &resistor\ r = \textbf{sigrel}\ (p, n)\ \textbf{where} \\ &\quad twoPin \diamond (p, n, u) \\ &\quad r \cdot p.i = u \end{aligned}$$

Encapsulated equation system fragment.

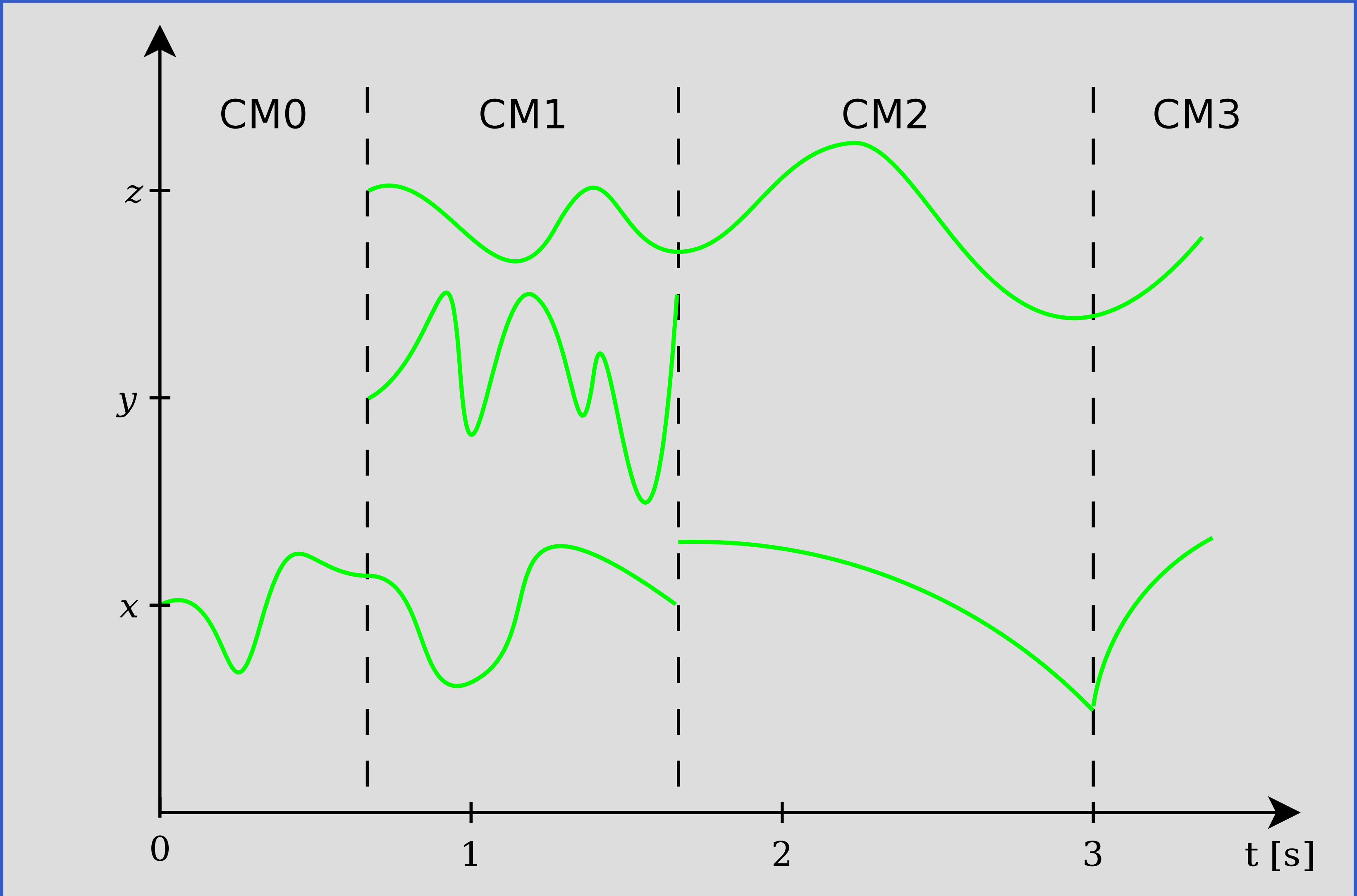
Signal relation ***application*** allows modular construction of models from component models.

The FHM Setting (4)

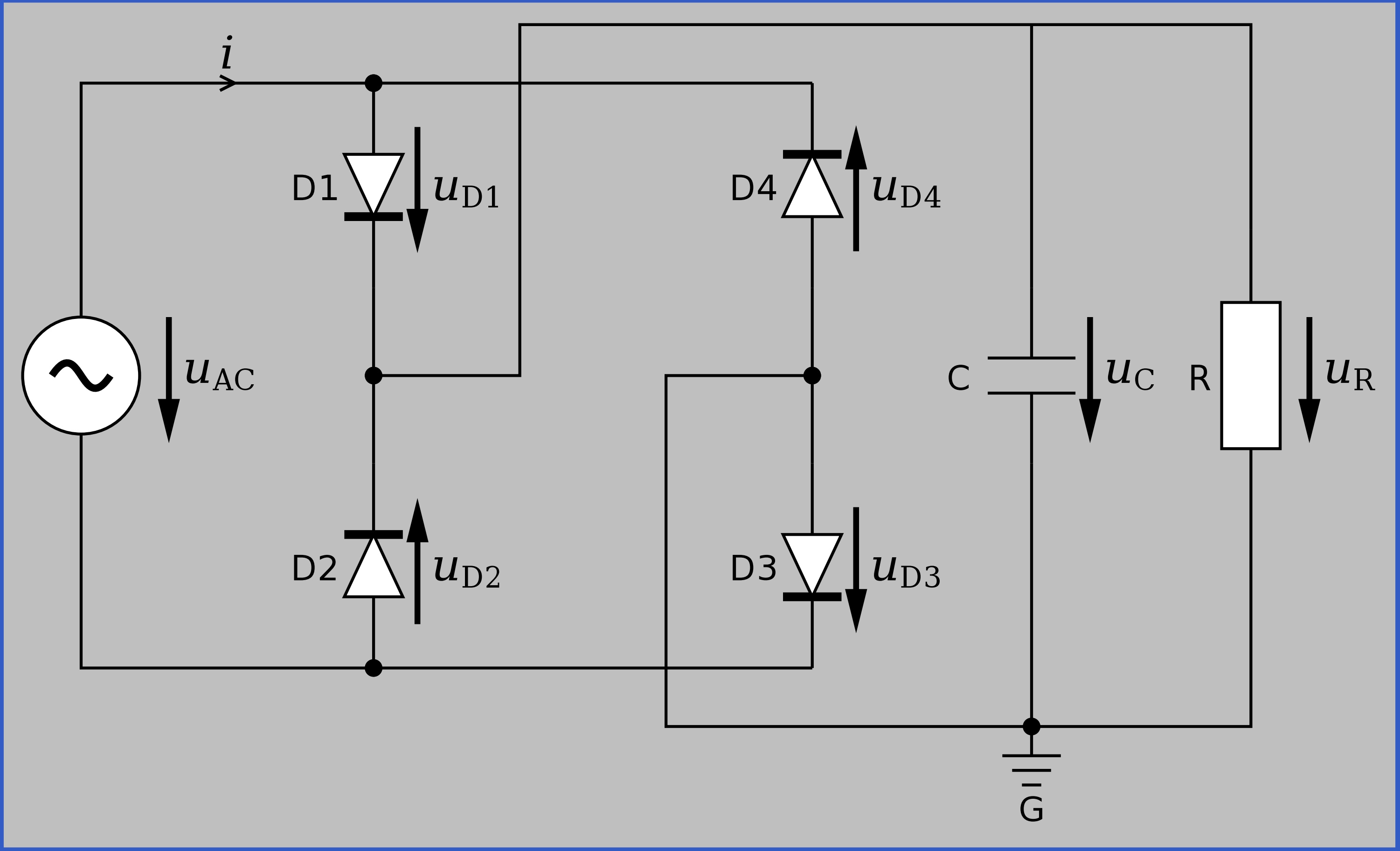
FHM is thus characterised by ***iterative staging***:

1. Compute model ("flat" system of equations)
2. Simulate (solve)
3. Repeat

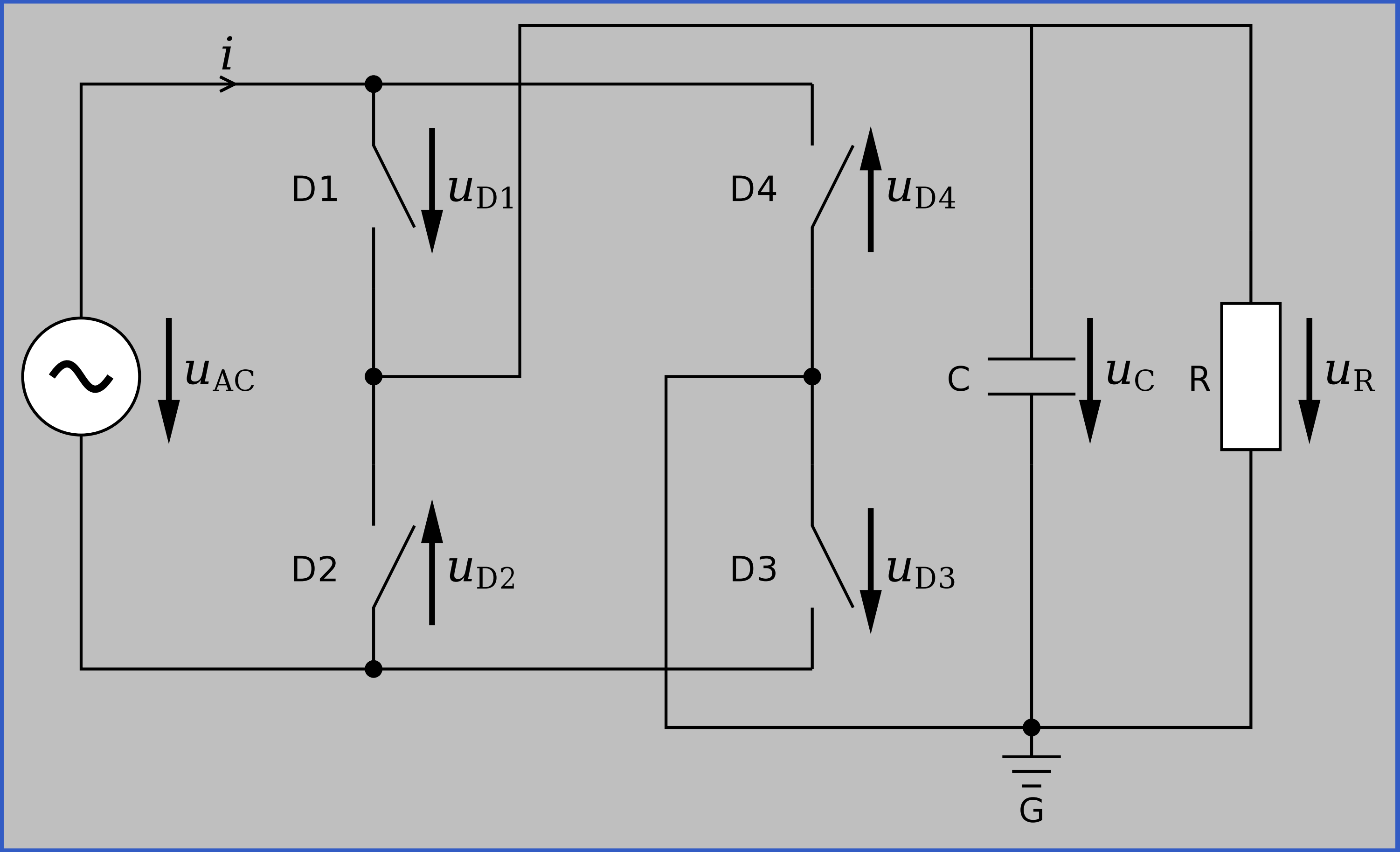
The FHM Setting (5)



Example: Ideal Diodes (1)



Example: Ideal Diodes (2)



The Problem (1)

We would like to know:

The Problem (1)

We would like to know:

- Are the computed systems of equations guaranteed to be solvable?

The Problem (1)

We would like to know:

- Are the computed systems of equations guaranteed to be solvable?
- Can each individual equation system fragment in principle form part of a solvable system?

The Problem (2)

- Consider:

$$x + y + z = 0 \tag{1}$$

The Problem (2)

- Consider:

$$x + y + z = 0 \tag{1}$$

  - Does not have a (unique) solution.

The Problem (2)

- Consider:

$$x + y + z \;=\; 0 \tag{1}$$

  - Does not have a (unique) solution.
  - Could be part of a system that does have a (unique) solution.

The Problem (2)

- Consider:

$$x + y + z = 0 \tag{1}$$

  - Does not have a (unique) solution.
  - Could be part of a system that does have a (unique) solution.
- The same holds for:

$$\begin{aligned} x - y + z &= 1 \\ z &= 2 \end{aligned} \tag{2}$$

The Problem (3)

- Composing (1) and (2):

$$\begin{aligned} x + y + z &= 0 \\ x - y + z &= 1 \\ z &= 2 \end{aligned} \tag{3}$$

Does have a solution.

The Problem (3)

- Composing (1) and (2):

$$\begin{aligned} x + y + z &= 0 \\ x - y + z &= 1 \\ z &= 2 \end{aligned} \tag{3}$$

Does have a solution.

- However, the following fragment is over-constrained:

$$\begin{aligned} x &= 1 \\ x &= 2 \end{aligned} \tag{4}$$

The Problem (4)

- Cannot answer questions regarding solvability comprehensively before we have a ***complete*** system. ***Inherently not very modular!***

The Problem (4)

- Cannot answer questions regarding solvability comprehensively before we have a ***complete*** system. ***Inherently not very modular!***
- Often ***undecidable*** whether a system of equations has a solution or not anyway.

The Problem (4)

- Cannot answer questions regarding solvability comprehensively before we have a ***complete*** system. ***Inherently not very modular!***
- Often ***undecidable*** whether a system of equations has a solution or not anyway.
- However, maybe violations of certain structural constraints related to solvability can be checked modularly? Two common examples:

The Problem (4)

- Cannot answer questions regarding solvability comprehensively before we have a ***complete*** system. ***Inherently not very modular!***
- Often ***undecidable*** whether a system of equations has a solution or not anyway.
- However, maybe violations of certain structural constraints related to solvability can be checked modularly? Two common examples:
  - ***Structurally non-singular system***

The Problem (4)

- Cannot answer questions regarding solvability comprehensively before we have a ***complete*** system. ***Inherently not very modular!***
- Often ***undecidable*** whether a system of equations has a solution or not anyway.
- However, maybe violations of certain structural constraints related to solvability can be checked modularly? Two common examples:
  - ***Structurally non-singular system***
  - ***Equation-variable balance***

Structural Non-singularity (1)

A system of equations is ***structurally singular*** iff not possible to put variables and equations in a one-to-one correspondence such that each variable occurs in the equation it is related to.

Structural Non-singularity (1)

A system of equations is ***structurally singular*** iff not possible to put variables and equations in a one-to-one correspondence such that each variable occurs in the equation it is related to.

- Structural non-singularity is neither a necessary nor sufficient condition for solvability.

Structural Non-singularity (1)

A system of equations is ***structurally singular*** iff not possible to put variables and equations in a one-to-one correspondence such that each variable occurs in the equation it is related to.

- Structural non-singularity is neither a necessary nor sufficient condition for solvability.
- However typical solvers are predicated on the system being structurally non-singular.

Structural Non-singularity (1)

A system of equations is ***structurally singular*** iff not possible to put variables and equations in a one-to-one correspondence such that each variable occurs in the equation it is related to.

- Structural non-singularity is neither a necessary nor sufficient condition for solvability.
- However typical solvers are predicated on the system being structurally non-singular.
- Insisting on structural non-singularity thus makes sense and is not overly onerous.

Structural Non-singularity (2)

Structural singularities can be discovered by studying the ***incidence matrix***:

Equations | Incidence Matrix

$$\begin{aligned} f_1(x, y, z) &= 0 \\ f_2(z) &= 0 \\ f_3(z) &= 0 \end{aligned}$$

$$\begin{array}{c} \begin{array}{ccc} x & y & z \end{array} \\ \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \end{array}$$

Structural Non-singularity (3)

Maybe we can annotate signal relation types with incidence matrices?

$$foo :: SR\,(Real, Real, Real) \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

$$\begin{aligned} & foo = \mathbf{sigrel}\ (x_1, x_2, x_3)\ \mathbf{where} \\ & \quad f_1\ x_1\ x_2\ x_3 = 0 \\ & \quad f_2\ x_2\ x_3 \quad\;\; = 0 \end{aligned}$$

Structural Non-singularity (3)

Maybe we can annotate signal relation types with incidence matrices?

$$foo :: SR\,(Real, Real, Real) \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

$$\begin{aligned} &foo = \mathbf{sigrel}\,(x_1, x_2, x_3)\ \mathbf{where} \\ &\quad f_1\ x_1\ x_2\ x_3 = 0 \\ &\quad f_2\ x_2\ x_3 = 0 \end{aligned}$$

Yes, possible, but rather complicated and we need to approximate anyway.

Equation-Variable Balance

Unequal number of equations and variables ***implies*** structurally singular system.

Maybe we can annotate signal relation types with ***balance***?

$$SR\,(\ldots)\ \boldsymbol{n}$$

Equation-Variable Balance

Unequal number of equations and variables ***implies*** structurally singular system.

Maybe we can annotate signal relation types with ***balance***?

$$SR\ (\ldots)\ \boldsymbol{n}$$

Yes, possible. And preservation holds! (Joey)

Equation-Variable Balance

Unequal number of equations and variables ***implies*** structurally singular system.

Maybe we can annotate signal relation types with ***balance***?

$$SR\ (\ldots)\ \boldsymbol{n}$$

Yes, possible. And preservation holds! (Joey)

But very weak assurances:

$$\begin{aligned} f(x, y, z) &= 0 \\ g(z) &= 0 \\ h(z) &= 0 \end{aligned}$$

Structural Well-Formedness (1)

In *Structural Types for Systems of Equations* we develop a notion of structural well-formedness for modular systems of equations that is:

Structural Well-Formedness (1)

In *Structural Types for Systems of Equations* we develop a notion of structural well-formedness for modular systems of equations that is:

- a better approximation of structural non-singularity than variable balance;

Structural Well-Formedness (1)

In *Structural Types for Systems of Equations* we develop a notion of structural well-formedness for modular systems of equations that is:

- a better approximation of structural non-singularity than variable balance;
- less precise, but much more practical (here) than the incidence-matrix-based approach.

Structural Well-Formedness (1)

In *Structural Types for Systems of Equations* we develop a notion of structural well-formedness for modular systems of equations that is:

- a better approximation of structural non-singularity than variable balance;
- less precise, but much more practical (here) than the incidence-matrix-based approach.

Signal relation types are annotated with ***balance***, the number of contributed equations, and types generally carry ***constraints*** on balance variables:

$$(2 \leq m \leq 4, 3 \leq n \leq 5) \Rightarrow SR\,(\ldots)\,m \rightarrow SR\,(\ldots)\,n$$

Structural Well-Formedness (2)

The approach distinguishes between two kinds of variables (number of each kind within parentheses):

- ***interface variables*** (i_Z)
- ***local variables*** (l_Z)

and three kinds of equations:

- ***interface equations*** (i_Q)
- ***mixed equations*** (m_Q)
- ***local equations*** (l_Q)

Total number of equations: $a_Q = i_Q + m_Q + l_Q$.

Structural Well-Formedness (3)

A signal relation is ***structurally well-formed*** (SWF) iff:

1. $l_Q + m_Q \geq l_Z$
2. $l_Q \leq l_Z$
3. $i_Q \leq i_Z$
4. $a_Q - l_Z \leq i_Z$
5. $i_Q \geq 0$, $m_Q \geq 0$, $l_Q \geq 0$

The balance (contribution) of a SWF relation is $n = a_Q - l_Z$.

Structural Dynamism

FHM allows for an evolving system of equations by switching blocks of equations in and out:

$$\textbf{initially}\ [\,;\textbf{when}\ \mathit{condition}_1\,] \Rightarrow$$
$$\quad \mathit{equations}_1$$
$$\textbf{when}\ \mathit{condition}_2 \Rightarrow$$
$$\quad \mathit{equations}_2$$
$$\ldots$$
$$\textbf{when}\ \mathit{condition}_n \Rightarrow$$
$$\quad \mathit{equations}_n$$

What about structural well-formedness?

Structural Dynamism and SWF (1)

Exactly one switch-branch active at any point.
How should the number of equations in each branch be related?

Structural Dynamism and SWF (1)

Exactly one switch-branch active at any point. How should the number of equations in each branch be related?

- ***Strong Approach***: exactly the same number of interface, mixed, and local equations in each branch.

Structural Dynamism and SWF (1)

Exactly one switch-branch active at any point.
How should the number of equations in each branch be related?

- ***Strong Approach***: exactly the same number of interface, mixed, and local equations in each branch.
  Very restrictive.

Structural Dynamism and SWF (1)

Exactly one switch-branch active at any point.
How should the number of equations in each branch be related?

- ***Strong Approach***: exactly the same number of interface, mixed, and local equations in each branch.
  Very restrictive.
- ***Weak Approach***: same number of equations in each branch.

Structural Dynamism and SWF (1)

Exactly one switch-branch active at any point. How should the number of equations in each branch be related?

- ***Strong Approach***: exactly the same number of interface, mixed, and local equations in each branch.
  Very restrictive.
- ***Weak Approach***: same number of equations in each branch.
  Loses too much information.

Structural Dynamism and SWF (1)

Exactly one switch-branch active at any point.
How should the number of equations in each branch be related?

- ***Strong Approach***: exactly the same number of interface, mixed, and local equations in each branch.
  Very restrictive.
- ***Weak Approach***: same number of equations in each branch.
  Loses too much information.
- ***Fair Approach***: branches are reconcilable.

Structural Dynamism and SWF (2)

A switch-block is ***reconcilable***, contributing i interface equations, m mixed equations, l local equations, iff i, m, l satisfying the following constraints for each branch k can be found:

6. $i \geq i_k \geq 0$
7. $l \geq l_k \geq 0$
8. $m \leq m_k - (i - i_k) - (l - l_k)$
9. $i + m + l = i_k + m_k + l_k$

Note: Interestingly, m may be negative!

Why No Preservation? (1)

Consider:

$$
\begin{aligned}
& \mathit{foo} = \textbf{sigrel}\ (x, y)\ \textbf{where} \\
& \qquad \textbf{local}\ z \\
& \qquad f\ (x, y, z) = 0 \\
& \qquad g\ (x) = 0 \\
& \mathit{fie} = \textbf{sigrel}\ (u)\ \textbf{where} \\
& \qquad \textbf{local}\ v \\
& \qquad \mathit{foo} \diamond (u, v)
\end{aligned}
$$

Why No Preservation? (1)

Consider:

$$
\begin{aligned}
&\mathit{foo} = \textbf{sigrel}\ (x, y)\ \textbf{where}\\
&\qquad \textbf{local}\ z\\
&\qquad f\ (x, y, z) = 0\\
&\qquad g\ (x) = 0\\
&\mathit{fie} = \textbf{sigrel}\ (u)\ \textbf{where}\\
&\qquad \textbf{local}\ v\\
&\qquad \mathit{foo} \diamond (u, v)
\end{aligned}
$$

Both foo and fie are structurally well-formed (why?) with balance 1 and 0, respectively.

Why No Preservation? (2)

But if we carry out some “flattening”:

$$
\begin{aligned}
&\mathit{fie} = \textbf{sigrel}\ (u)\ \textbf{where}\\
&\quad \textbf{local}\ z, v\\
&\quad f\ (u, v, z) = 0\\
&\quad g\ (u) = 0
\end{aligned}
$$

Why No Preservation? (2)

But if we carry out some “flattening”:

$$
\begin{aligned}
&\mathit{fie} = \textbf{sigrel}\ (u)\ \textbf{where} \\
&\quad \textbf{local}\ z, v \\
&\quad f\ (u, v, z) = 0 \\
&\quad g\ (u) = 0
\end{aligned}
$$

The equation that initially was classified as mixed turned out to be an interface equation; only one equation to solve for two local variables z and v.

Why No Preservation? (2)

But if we carry out some “flattening”:

$$
\begin{aligned}
&\mathit{fie} = \textbf{sigrel}\ (u)\ \textbf{where}\\
&\quad \textbf{local}\ z, v\\
&\quad f\ (u, v, z) = 0\\
&\quad g\ (u) = 0
\end{aligned}
$$

The equation that initially was classified as mixed turned out to be an interface equation; only one equation to solve for two local variables z and v.

Reduction turned a structurally well-formed relation into one that is ill-formed.

Correctness properties?

The following seems plausible:

> If $t : C \Rightarrow SR\;()\;n$ and $\neg\mathrm{satisfiable}(C)$, then there exists a structural configuration (i.e., a particular choice of switch branches), such that t elaborates to a structurally singular system of equations.

We have not yet attempted to prove this.