

A Hoare Logic for Single-Input Single-Output Continuous-Time Control Systems

June 2002

Richard Boulton

School of Computer Science
University of St Andrews

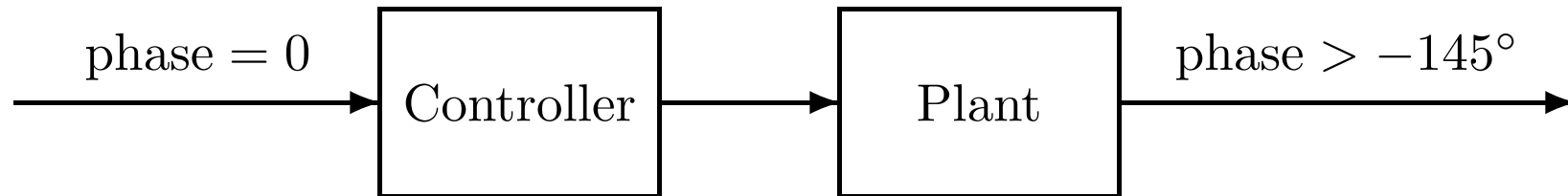
North Haugh
St Andrews
Fife KY16 9SS
Scotland, UK

Work funded by QinetiQ

Motivation

- Aim to adapt symbolic analysis techniques used for software to Simulink
- Need to cast problem in form to which these techniques apply
 - Make Simulink diagrams have similar shape to programs?
 - Identify properties that compose with “program” constructs
- Hoare Logic: an early and widely used technique
- This talk:
 - Properties of gain and phase
 - Building blocks of control systems
 - Introduction to Predicate Logic and Hoare Logic
 - Hoare logic for control systems and its automation

Annotating Control Systems with Properties



In Hoare Logic, it would look something like this:

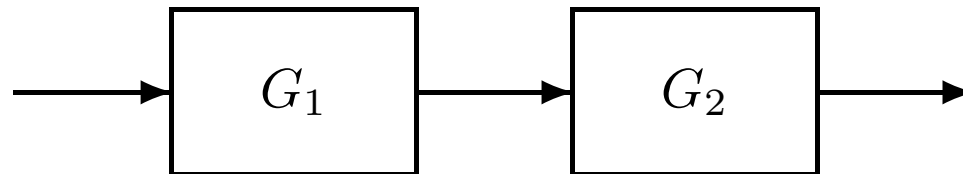
$$\{\theta = 0\} \text{Controller}; \text{Plant} \{\theta > -145^\circ\}$$

Features:

- Properties can contain symbolic parameters
 - can show specification met for a range of parameter values
- Can use logical reasoning

The Key Observation

Gain and phase compose for sinusoidal signals



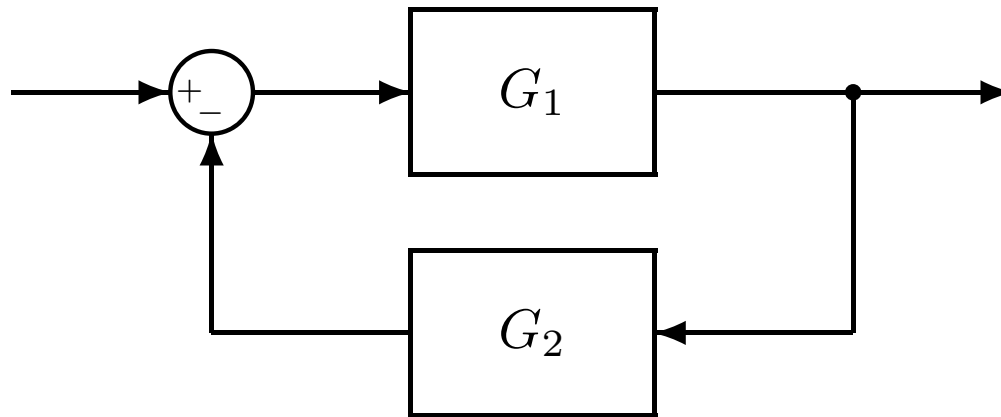
$$G_1(j\omega) = r_1 e^{j\theta_1}$$

$$G_2(j\omega) = r_2 e^{j\theta_2}$$

$$G_1(j\omega)G_2(j\omega) = (r_1 r_2) e^{j(\theta_1 + \theta_2)}$$

Closed Loops

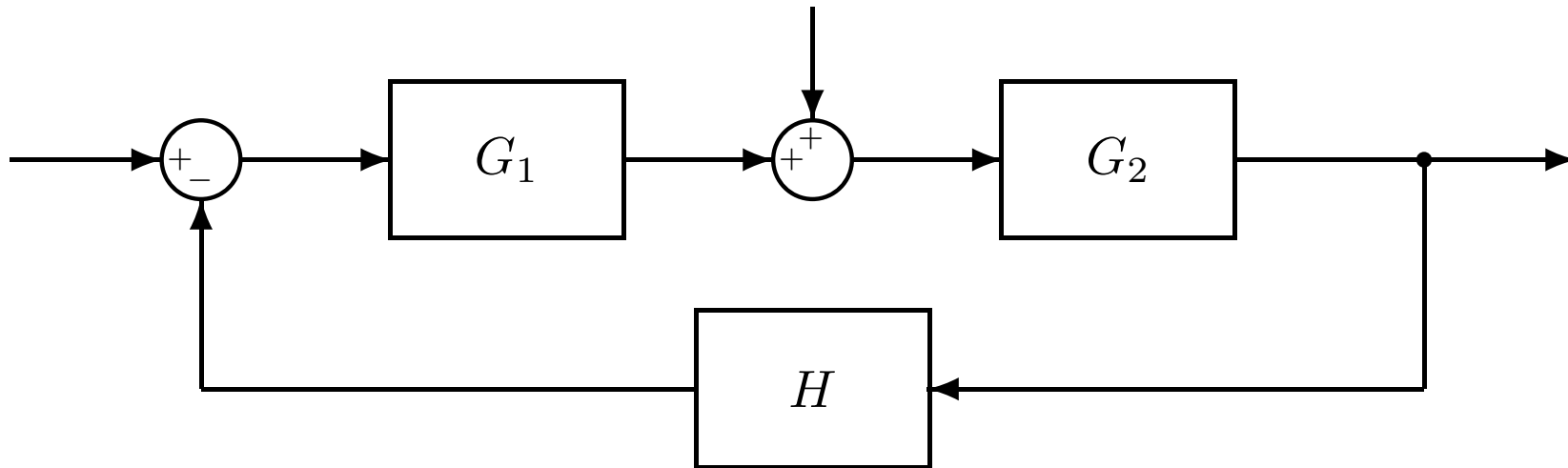
But do gain and phase compose for feedback loops?



Yes, but they become inter-dependent, e.g.

$$\text{gain of loop} = \frac{r_1}{\sqrt{r_1^2 r_2^2 + 2r_1 r_2 \cos(\theta_1 + \theta_2) + 1}}$$

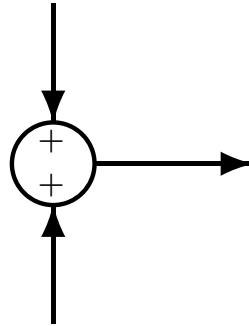
More Complex Systems



What structures are there in addition to sequences and loops?

We want all compound structures to be a composition of one or more simpler structures

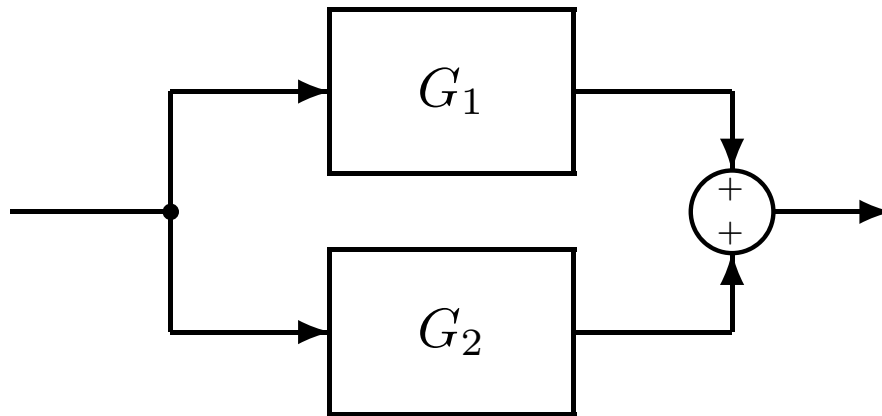
Summation



How do we express this as a composition of components?

Summation (cont.)

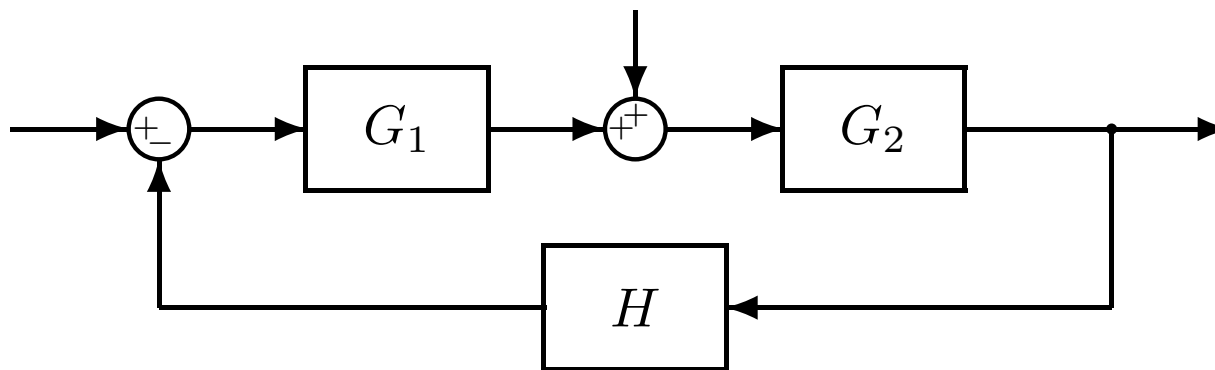
Assume that for single-input systems the two signals to be summed ultimately come from the same source:



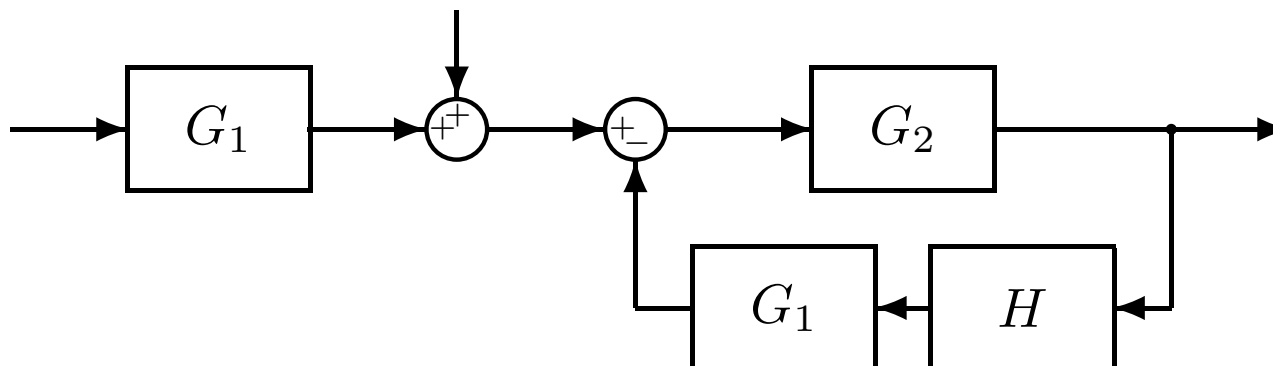
G_1 and G_2 may be large subsystems

The Complex System Again

Intertwined structures: need to make a transformation



becomes



Duplication of subsystems OK for mathematical analysis?

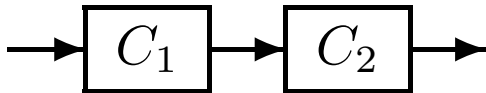
A Complete Set of Building Blocks?



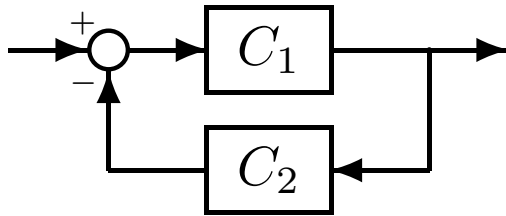
Unit



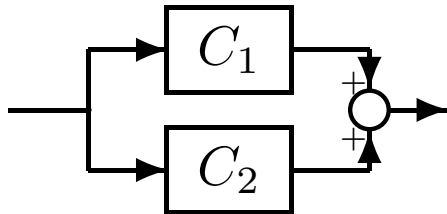
Fcn($dr, d\theta$)



Seq(C_1, C_2)



Loop(C_1, C_2)

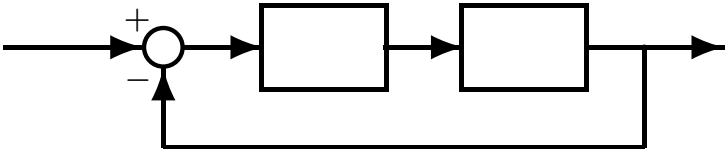


Sum(C_1, C_2)

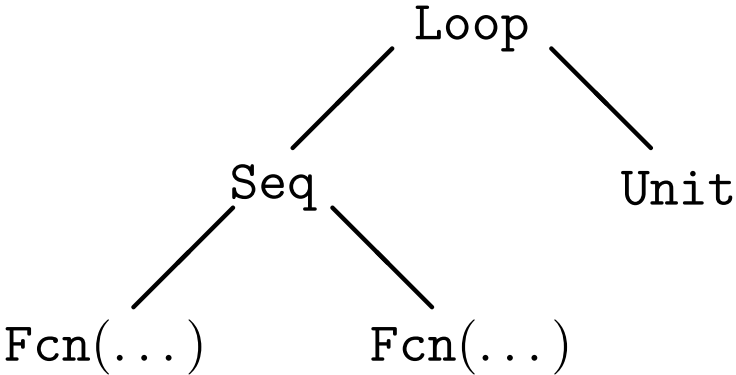
C_1 and C_2 are arbitrary subsystems

Trees

Control systems formed from the building blocks can be expressed as trees, e.g.



can be written as



i.e.

$$\text{Loop}(\text{Seq}(\text{Fcn}(\dots), \text{Fcn}(\dots)), \text{Unit})$$

This aids logical/mathematical analysis

Arbitrary Systems

- Simulink diagrams are *graph* structures, not trees
- Our intention is to transform them into trees
- But is this always possible?
 - Not sure
 - but the building blocks can express a large range of diagrams
- Have satisfied two criteria stated earlier:
 - Control systems formed from building blocks have similar shape to programs
 - Properties compose with constructs (the building blocks)

Predicate Logic

- Also known as *Predicate Calculus* or *First-Order Logic*
- Atomic formulae are applications of predicates
 - e.g. $Brown(x)$ can denote that entity x has colour brown
- The result of applying a predicate is always a Boolean value: true or false
- Arithmetic relations can also be viewed as predicates
 - e.g. ' $<$ ' is a predicate that takes two arguments, both of which are numbers
 - $x < 1$ might be written $Less(x, 1)$ in more general notation

Logical Operators (Connectives)

Atomic formulae of predicate logic can be combined using the usual logical operators:

\neg	logical NOT	negation
\wedge	logical AND	conjunction
\vee	logical OR	disjunction
\Rightarrow		implication

e.g.

$$(x < y) \wedge (y < 2) \Rightarrow (x < 2)$$

Implications can be read as “if ... then ...”

By default, a formula states a property for all possible values of the variables (unknowns)

Hoare Logic

- A Hoare logic can be used to give a mathematical meaning to a formal language
- Hoare's original logic was for a simple imperative programming language in the style of Pascal
- An alternative to execution or simulation for analysing programs
- It uses *preconditions* and *postconditions* to assert logical properties before and after evaluation of an expression of the language
- Preconditions and postconditions are predicate logic formulae
- Example:

$$\{X = 1\}X := X + 1\{X = 2\}$$

Axioms and Rules

- Statements in Hoare Logic may be true or false
- The aim is to deduce a true statement about a program using
 - simple statements that are known to be true (axioms), and
 - rules that deduce a true statement from other true statements

The Assignment Axiom

$$\vdash \{P[X \setminus e]\}X := e\{P\}$$

e.g.

$$\vdash \{P[X \setminus X + 1]\}X := X + 1\{P\}$$

For a particular property P :

$$\vdash \{(X = 2)[X \setminus X + 1]\}X := X + 1\{X = 2\}$$

$$\vdash \{(X + 1 = 2)\}X := X + 1\{X = 2\}$$

$$\vdash \{(X = 2 - 1)\}X := X + 1\{X = 2\}$$

$$\vdash \{(X = 1)\}X := X + 1\{X = 2\}$$

The Sequencing Rule

$$\frac{\vdash \{P\}C_1\{Q\} \quad \vdash \{Q\}C_2\{R\}}{\vdash \{P\}C_1;C_2\{R\}}$$

e.g.

$$\frac{\vdash \{X = 1\}X := X + 1\{X = 2\} \quad \vdash \{X = 2\}X := X * 2\{X = 4\}}{\vdash \{X = 1\}X := X + 1; X := X * 2\{X = 4\}}$$

Note that the rule holds for all programs C_1 and C_2 , and for all properties (predicates) P , Q , and R

Important Feature

- We are not restricted to deducing (proving) the strongest possible property
 - e.g. instead of $X = 4$, we might prove $X < 6$
 - This gives us some flexibility

A Hoare Logic for Control Systems

- In the original Hoare logic, the argument of a predicate is a *program state* containing the values of all the program variables
- Want to adapt Hoare logic to control systems
 - For frequency response, predicates operate over gain and phase
 - Preconditions and postconditions assert properties of gain and phase at input and output of a component or subsystem
- Also need to maintain expressions for the gain and phase
- The constructs of our “language” are the building blocks: Unit, Fcn, Seq, Loop, and Sum

Statements in the Logic

$$\{P\}C\langle dr, d\theta\rangle\{Q\}$$

This means component C (be it atomic or compound) causes a gain of dr and a phase shift of $d\theta$, and if property P holds at the input, then property Q holds at the output

Within properties

- r denotes the gain at the point of assertion
- θ denotes the phase

Restricted to single-input single-output systems

Some Axioms and Rules

The (Transfer) Function Axiom

$$\frac{}{\vdash \{P[r \setminus r * dr, \theta \setminus \theta + d\theta]\} \text{Fcn}(dr, d\theta) \langle dr, d\theta \rangle \{P\}}$$

The Sequencing Rule

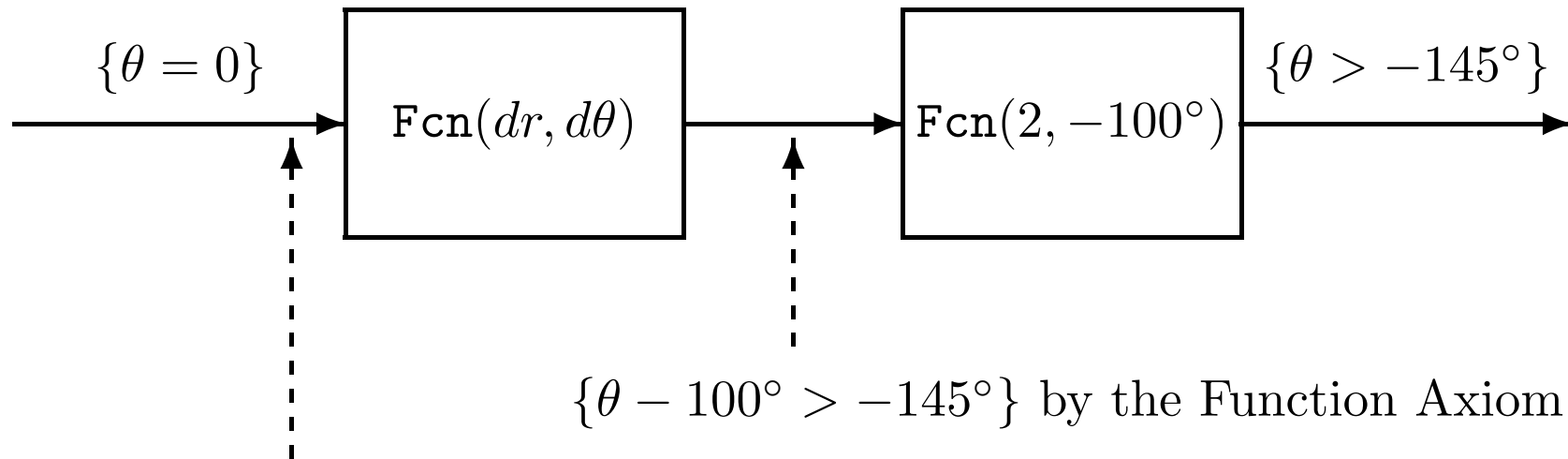
$$\frac{\vdash \{P\}C_1 \langle dr_1, d\theta_1 \rangle \{Q\} \quad \vdash \{Q\}C_2 \langle dr_2, d\theta_2 \rangle \{R\}}{\vdash \{P\} \text{Seq}(C_1, C_2) \langle dr_1 * dr_2, d\theta_1 + d\theta_2 \rangle \{R\}}$$

Precondition Strengthening

$$\frac{\vdash P' \Rightarrow P \quad \vdash \{P\}C \langle dr, d\theta \rangle \{Q\}}{\vdash \{P'\}C \langle dr, d\theta \rangle \{Q\}}$$

Example System with Properties

$$\{\theta = 0\} \text{Seq}(\text{Fcn}(dr, d\theta), \text{Fcn}(2, -100^\circ)) \langle \dots \rangle \{\theta > -145^\circ\}$$



$\{(\theta + d\theta) - 100^\circ > -145^\circ\}$ by the Function Axiom (again)

Proof About Example System

Th1: $\vdash \{\theta - 100^\circ > -145^\circ\} \text{Fcn}(2, -100^\circ) < 2, -100^\circ > \{\theta > -145^\circ\}$

Th2:

$\vdash \{(\theta + d\theta) - 100^\circ > -145^\circ\} \text{Fcn}(dr, d\theta) < dr, d\theta > \{\theta - 100^\circ > -145^\circ\}$

Th3: $\vdash \{(\theta + d\theta) - 100^\circ > -145^\circ\}$

$\text{Seq}(\text{Fcn}(dr, d\theta), \text{Fcn}(2, -100^\circ)) < dr * 2, d\theta - 100^\circ >$

$\{\theta > -145^\circ\}$

Now, if we can prove $(\theta = 0) \Rightarrow ((\theta + d\theta) - 100^\circ > -145^\circ)$, can use precondition strengthening to get:

Th4: $\vdash \{\theta = 0\}$

$\text{Seq}(\text{Fcn}(dr, d\theta), \text{Fcn}(2, -100^\circ)) < dr * 2, d\theta - 100^\circ >$

$\{\theta > -145^\circ\}$

Example Proof (cont.)

- Can only prove the implication if $d\theta > -45^\circ$
 - a constraint on the controller $\text{Fcn}(dr, d\theta)$
- Could also use value from $\langle dr * 2, d\theta - 100^\circ \rangle$
 - but that's not always possible
- For realistic examples, phase and gain of Fcn 's would be functions of frequency
 - Then prove property for all frequencies
 - Or constrain frequency, e.g. $0.1 < \omega \Rightarrow \dots$
 - Example: the transfer function $\frac{1}{s+1}$ is represented as:

$$\text{Fcn} \left(\frac{1}{\sqrt{1 + \omega^2}}, \arctan(-\omega) \right)$$

Possible Rule for Loops

$$\frac{\vdash \{P\}C_1 \langle dr_1, d\theta_1 \rangle \{Q\} \quad \vdash \{Q\}C_2 \langle dr_2, d\theta_2 \rangle \{R\}}{\vdash \{P[r \setminus r * lr(dr_1, d\theta_1, dr_2, d\theta_2), \theta \setminus \theta + lt(dr_1, d\theta_1, dr_2, d\theta_2)]\} \\ \text{Loop}(C_1, C_2) \langle lr(dr_1, d\theta_1, dr_2, d\theta_2), lt(dr_1, d\theta_1, dr_2, d\theta_2) \rangle \\ \{R[r \setminus r * (dr_1 * dr_2), \theta \setminus \theta + (d\theta_1 + d\theta_2)]\}}$$

where

$$lr(dr_1, d\theta_1, dr_2, d\theta_2) = \frac{dr_1}{\sqrt{dr_1^2 dr_2^2 + 2dr_1 dr_2 \cos(d\theta_1 + d\theta_2) + 1}}$$

$$lt(dr_1, d\theta_1, dr_2, d\theta_2) = \arctan \left(\frac{\sin d\theta_1 - dr_1 dr_2 \sin d\theta_2}{\cos d\theta_1 + dr_1 dr_2 \cos d\theta_2} \right)$$

Automation

- The Hoare logic rules are difficult to use
- Rules can be formalised in a theorem proving system
- Automatic tools can decompose a statement of Hoare logic
 - Produces the set of predicate logic formulae necessary to prove original statement
 - These formulae are called *verification conditions* (VCs)
 - We have a prototype of such a tool for our Hoare logic
- Similar process used to verify implementation of a control system (Ada code) w.r.t. Z specifications generated by ClawZ

Uses of the Hoare Logic

- Calculation of gain and phase, and controller parameters
- Proving constraints on gain and phase
 - at a particular frequency, for a range of frequencies, for all frequencies
 - including properties that cannot be expressed graphically
- Deriving symbolic constraints on controller parameters
- New ways of analysing closed loops?
- Properties of subsystems could be reused
- Proving “correctness” of transformations on Simulink diagrams
 - preserve gain and phase
- *A framework for analysing frequency response*

Discrete-Time Control Systems

- Gain and phase relationships appear to be same as in continuous-time
- But:
 - Gain and phase of atomic components will be different
 - Mixing discrete and continuous components would complicate matters
 - Should be OK if treat hold + plant as a unit
 - Theory breaks down for high frequencies — need constraints

Conclusions

- First steps towards exploiting composability in proofs
- Rules give insight into behaviour of components for sinusoidal inputs
- Parameters of controller (and plant) can be symbolic
- Properties can also involve symbolic parameters
- Many ways in which the Hoare logic could be used
- Rules have been mechanised and a prototype VC generator produced
- Currently limited to single-input single-output

Future Work

- Refine prototype and test on larger examples
- Investigate the transformation of Simulink diagrams into our building blocks
- Investigate how assertions can be included in Simulink diagrams
- Discrete-time systems
- Multiple input/output systems???