

Towards a Hoare Logic for Continuous-Time Control Systems*

Richard J. Boulton, Ruth Hardy, Ursula Martin
School of Computer Science
University of St Andrews

25 October 2002

Abstract

This paper presents a Hoare-style logic for reasoning about the frequency response of control systems in the continuous-time domain. Two properties, the gain (amplitude) and phase shift, of a control system are considered. These properties are for a sinusoidal input of variable frequency.

1 Introduction

Many man-made dynamical systems such as cars, planes, CD-players and nuclear reactors are augmented with a control system in hardware or software. The physical system is typically referred to as the *plant*. Some plants are inherently unstable in the absence of a control system (e.g. many fighter aircraft), and the systems are often safety critical or mission critical.

Most control systems are configured as a *closed loop* (a feedback loop) in which the outputs or current behavior of the plant are measured and subtracted from a reference input (a control value such as desired cruising speed of a car). The resulting difference is used as input to a controller which in turn produces signals to control the plant. There may also be a control component in the feedback path. Thus, a typical control system has the form illustrated in Figure 1.

Control systems may be modelled in the continuous time domain or in the discrete time domain. In the former, signals are continuously varying

*Research supported by QinetiQ and by an EPSRC studentship to the second author.

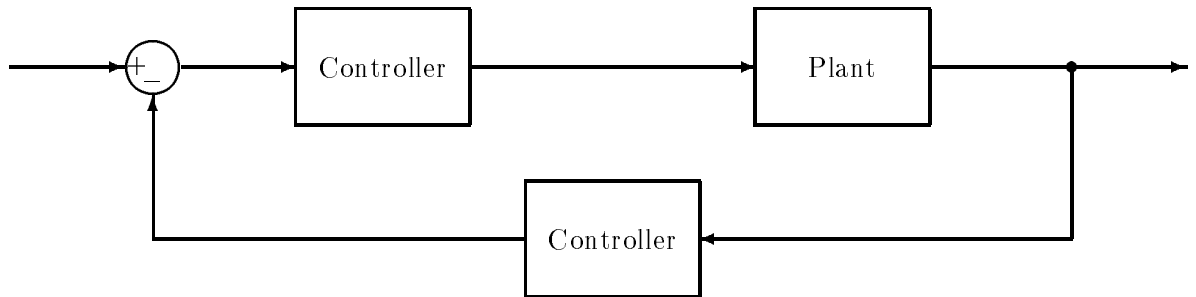


Figure 1: Typical shape of a control system

with time modelled as a real number. In the latter, the signals are sampled at discrete time intervals and so the value of a signal may be discontinuous and time can be modelled using integers. A discrete-time signal can be related to continuous-time signals by *holding* the sampled value constant until the next sample is taken. A discrete-time model is required for implementation using a digital circuit or software. It is common for a continuous-time model to be developed initially, which is then adapted to the discrete domain prior to implementation.

Numerical modelling, simulation, and analysis of control systems are supported by computer software. For example Mathworks Simulink [13] provides a graphical representation of a control system as the standard engineers' block-diagram, which is obtained as the Laplace transform of the original dynamical system.

While there is a wealth of academic literature on the design of control systems [16], less attention has been paid to design validation, especially of software systems. In practice visual inspection of numeric plots is widely used: for example, suites of Bode and Nichols plots are used to specify and discharge design requirements for flight control [17], expressed in terms of phase and gain of an input signal.

The use of formal methods and computational logic in the analysis of control systems is of increasing importance, but has thus far largely been confined to hybrid systems and statechart-like models. The widespread use of Simulink suggests that effective formal verification techniques for block diagrams could have significant impact. In general terms one might expect to annotate points in a diagram with assertions stating what was true at that point, for example a property of phase or gain, and use a logic to reason

about the assertions. Thus, for example, one might hope to replace the plotting described above with an automated analysis using computational logic.

In a this study we develop a Hoare-style logic and a verification condition generator for a restricted class of block diagrams, essentially those with a tree structure. Hoare logics [23] were originally studied by Hoare, Floyd and others to give an axiomatic basis for programming, and continue to be used for a variety of applications, for example Java byte-code verification [24]. As far as we know our work is the first to investigate Hoare-style logics for feedback systems. We attached assertions to nodes in the diagram: the key observation was that phase and gain were compositional, and hence we could reason about them locally, and propagate our reasoning through the diagram to deduce properties of a classical frequency-response analysis. Following Gordon's approach [25] the logic has been mechanised in the HOL98 theorem proving system, allowing goal-directed reasoning, machine assistance in the details of the proof, and automatic generation of verification conditions, the logical formulas that must ultimately be proved to justify an assertion in the Hoare logic. The VCs themselves are pure predicate logic formulas, that is, they do not involve the constructs of our logic.

These ideas also enabled us to prototype an automated alternative to Nicholls plots [26], which replaced numeric plots with symbol manipulation in the computer algebra system Maple and the theorem prover PVS (we could also have used quantifier elimination). This in turn exploited Gottliebsens PVS continuity checker [27].

To simplify matters we chose initially to work with continuous-time rather than discrete models, with a single input and a single output. Since the behaviour of continuous-time models is simpler than that of discrete models, we have begun by investigating the former. Section 9 discusses briefly how the ideas might be adapted to the discrete time domain. We also restrict consideration to systems with a single input and a single output. To a certain extent, multi-input/multi-output systems can be forced into our restricted system by the use of multiplexing. A more general analysis would use state space models.

In practice the stability, time and frequency response of control systems are often analysed: we chose the latter, which treats the amplitude and phase shift of the output signal when the system is presented with a sinusoidal input with a range of frequencies. The key observation, on which the rest of the paper is built, is that gain (amplitude) and phase are properties that behave compositionally as larger control systems are constructed from subsystems. This allows us to build a logic for properties of control systems

in the style of Hoare logics.

In the rest of this paper we present gain and phase, our language Cosy for representing block diagrams, our Hoare logic, a simple worked example showing analysis of gain and phase, brief details of our implementation in a theorem proving system, and directions for further work.

2 Background and related work

Control engineering is a large subject: we intend to focus on those aspects which are to a control engineer fairly standard and widely used in practice [16]. Optimal control assumes that a model of the system is available and one wants to optimise its behaviour, using the calculus of variations and so forth: for example pre-computing a desired flight-path for a spacecraft. Feedback control compensates for uncertainty in the model by using feedback to correct for deviations for desired behaviour: for example if the spacecraft strays off course. Models vary according to the application: for example differential equations are used when modelling a continuous signal, but these are replaced by difference equations when modelling a sampled signal as used in digital systems. In reasoning about such systems we are interested not only in the solutions, but in their properties. These include the time response, stability, frequency response and behaviour under perturbation. The time response considers features such as the time taken for a property of the system (e.g. the cruising speed of a car) to reach the desired value, and by how far it overshoots before settling at the desired value. Stability analysis considers whether the system will always settle into a steady state following a change to the input(s). An output of an unstable system may increase out of control or oscillate. Frequency response considers the amplitude and phase shift of the output signal when the system is presented with a sinusoidal input. The analysis considers input signals with a range of frequencies.

In practice systems are rarely linear: non-linear systems are generally treated locally by linearising at points of interest, but global behaviour is subject of much research and raises subtle questions in differential and algebraic geometry.

In classical control a Laplace transform is applied to a linear system to obtain a representation as a transfer function, a rational function over the complexes. Analysis of properties, such as frequency and response of the control system, is in terms of the position of its roots and poles in the complex plane. So called modern control considers a state-space representation, which replaces a single differential equation with a system of simultaneous

equations in the state variables, and analyses the system via properties of the eigenvalues of a related matrix. Both frameworks can be extended from SISO (single input) to MIMO (multiple input) systems.

Block diagrams are often used to represent systems with feedback graphically, for example in classical control a block diagram is a directed graph whose edges are labelled by rational functions over the complexes. They also allow more general representation of components described only by their input/output behaviour.

Software such as the widely used Mathworks Simulink [13], the industry standard in avionics and automotive applications, supports numerical tests and simulations. A number of standard tests are used for prediction and analysis: for example the Nichols plot is a numerical test which investigates stability. It displays the steady state behaviour of a classical control system in terms of the phase and gain of a sinusoidal input. The control requirements of fighter aircraft are specified in terms of acceptable paths in this plot [17].

In practice man-made control systems are typically digital embedded software systems, which use sampled, rather than continuous time. These can be modelled as discrete dynamical systems (difference equations), which again admit a transform representation via the z-transform, and an analogous state-space representation, investigated as before using matrix algebra. The design of a digital controller, for example in avionics applications, typically involves analysis as above in continuous time: it is then passed to a software team for implementation as a discrete digital system. It has been suggested that this process is a likely source of error: indeed apparently similar continuous and discrete systems may have very different stability properties. The ubiquity of such embedded controllers, for example in cars and domestic appliances, has led to increased interest in methods of generating assured code straight from a high level design [3].

Our main focus is engineering applications: however biology provides a wealth of control mechanisms to explore. Here the process is reversed: we want to infer the model from knowledge of its properties. Typically in investigating transcriptional control (the cellular regulation of genes and proteins), microarray analysis is used to gather data on response to perturbations, for example in temperature or concentration of a substance.

Our thesis is that a control system can be regarded directly as a computational process, of which the various mathematical descriptions, whether of a continuous or discrete system, can be regarded as a high-level specification or representation. Thus it is natural to think of this process in logical form, and to extend and apply familiar computer science techniques to un-

derstand, model and reason about it. In general terms one might expect to annotate a representation, for example nodes in a block diagram, with assertions, and use a logic to reason about the assertions. Thus, for example, the numeric plotting described above can be viewed as an assertion about the output from a given real input to a complex function: one might hope to replace it with an automated analysis of properties of more general state variables.

The study of control in the context of computer science is an emerging area: we identify some strands of work which complement our own:

- The most well developed is the field of hybrid systems, which models certain control systems as automata with discrete transitions which are then amenable to model checking [11] and theoretical analysis [12]. Alur and Dill [2] introduced timed automata, state-transition diagrams annotated with timing constraints using finitely many real-valued clock variables which can be used to model discrete dynamical systems
- In the 1970s Arbib and Manes [1] studied categorical models of linear control: more recently various categories with feedback have been much studied, especially traced monoidal [9] categories, which are models for linear logic. These seem to obey similar algebraic laws to our feedback diagrams, though as far as we know the connection has not been developed formally. Turlas [19] has studied reasoning about general diagram languages.
- Less attention has been paid to the classical dynamical systems representations. Perhaps the closest foundational work is Edalats [6] extension of classical domain theory to analysis and dynamical systems. Tiwari [18] allows abstraction of dynamical systems to a level where model checking can be used.
- Our own work on light formal methods for mathematical systems [4, 5] was a precursor to this proposal: NAG Ltd funded us to devise an assertion language and lint-like checker for their AXIOM system.
- The widespread use of Simulink suggests that effective formal verification techniques for block diagrams could have significant impact. The Clawz system of Arthan et al [3], developed for Qinetiq, is a first step: it translates discrete-time models, described using Simulink, into formal specifications in Z. A controller implementation in an Ada-like programming language can then be verified against these Z specifications using the ProofPower mechanised proof assistant. Qinetiq used

Clawz in a case study of the braking systems of the Eurofighter, and are addressing issues of concurrency in this framework using CSP/FDR. Mahony [14] has used similar ideas in his DOVE system.

3 Composition of Gain and Phase

Most control engineering is done using Laplace transforms. A signal may be represented as a function f of time t . The Laplace transform maps f to a new function F that has a complex frequency value as its argument. The variable s is typically used to denote the argument. The Laplace transform is given by the following integral:

$$F(s) = \int_0^{\infty} f(t)e^{-st} dt$$

where $f(t)$ is assumed to be zero for negative values of t .

As well as mapping signals to their transforms, components in a control system are also represented using Laplace transforms. The Laplace transforms have nice properties, e.g. the transform for two components in sequence is the product of the transforms of the components. Similarly, the output signal of a component is represented by the product of the transform of the input signal and the transform of the component. Laplace transforms apply to the continuous time domain. A different kind of transform is used for discrete systems. For more about Laplace transforms, see any introductory text on control engineering, e.g. [16].

In frequency analysis, we are interested in the behaviour of sinusoidal signals. The analysis is done using Fourier transforms. It turns out that the Fourier transform can be obtained from the Laplace transform by substituting $j\omega$ for the variable s , where j is $\sqrt{-1}$ and ω is the (real-valued) frequency of the signal:¹

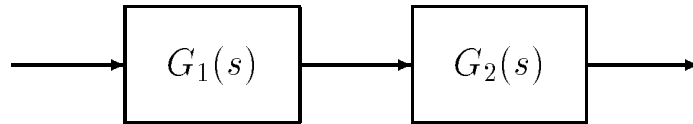
$$\begin{aligned} F(j\omega) &= \int_0^{\infty} f(t)e^{-j\omega t} dt \\ &= \int_{-\infty}^{\infty} f(t)e^{-j\omega t} dt \quad \text{because } f(t) = 0 \text{ for } t < 0 \end{aligned}$$

The Fourier transform gives us a complex number which we can write in the form $re^{j\theta}$. When the transform represents a sinusoidal signal, r is its amplitude and θ is its phase. When the transform represents a component in the control system, r is the gain (the factor by which the component increases the amplitude) and θ is the change in phase caused by the component.

¹Control engineers conventionally use j rather than i and we follow that practice here.

3.1 Sequencing of Components

Now suppose we have a control (sub)system constructed from two components in sequence:



where $G_1(s)$ and $G_2(s)$ are the Laplace transforms of the components. As stated above, the Laplace transform for the combined system is given by the product $G_1(s)G_2(s)$ of these two transforms. Hence, the Fourier transform is $G_1(j\omega)G_2(j\omega)$:

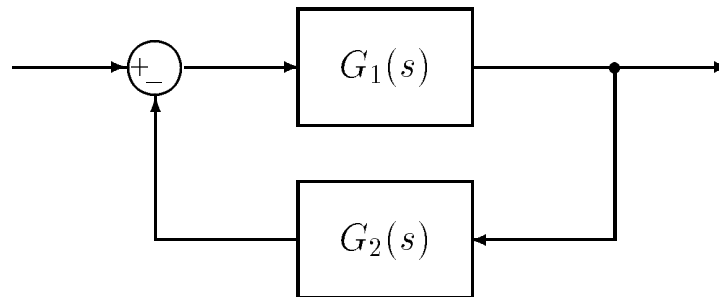
$$\begin{aligned} G_1(j\omega)G_2(j\omega) &= r_1 e^{j\theta_1} r_2 e^{j\theta_2} \\ &= (r_1 r_2) e^{j(\theta_1 + \theta_2)} \end{aligned}$$

From this it is clear that the gain of the combined system is the product of the constituent gains, and the phase shift of the combined system is the sum of the constituent phase shifts.

So, for sequencing of components, gain and phase compose in a straightforward manner. But, there are other structures to be found in the models of control systems, most notably feedback loops and summing points.

3.2 Feedback Loops

Consider the following closed loop system:



The Laplace transform for the entire closed loop system is:

$$C(s) = \frac{G_1(s)}{1 + G_1(s)G_2(s)}$$

and $G_1(j\omega)$ and $G_2(j\omega)$ are given by:

$$\begin{aligned} G_1(j\omega) &= r_1 e^{j\theta_1} = r_1 (\cos \theta_1 + j \sin \theta_1) \\ G_2(j\omega) &= r_2 e^{j\theta_2} = r_2 (\cos \theta_2 + j \sin \theta_2) \end{aligned}$$

So,

$$C(j\omega) = \frac{r_1 e^{j\theta_1}}{1 + r_1 e^{j\theta_1} r_2 e^{j\theta_2}} = \frac{r_1 e^{j\theta_1}}{1 + r_1 r_2 e^{j(\theta_1 + \theta_2)}}$$

The imaginary part of the denominator can be eliminated by multiplying top and bottom by the complex conjugate of the denominator:

$$\begin{aligned} C(j\omega) &= \frac{r_1 e^{j\theta_1} (1 + r_1 r_2 e^{-j(\theta_1 + \theta_2)})}{(1 + r_1 r_2 e^{j(\theta_1 + \theta_2)})(1 + r_1 r_2 e^{-j(\theta_1 + \theta_2)})} \\ &= \frac{r_1 (e^{j\theta_1} + r_1 r_2 e^{-j\theta_2})}{1 + 2r_1 r_2 \cosh(j(\theta_1 + \theta_2)) + r_1^2 r_2^2} \quad [e^\theta + e^{-\theta} = 2 \cosh \theta] \\ &= \frac{r_1 (e^{j\theta_1} + r_1 r_2 e^{-j\theta_2})}{1 + 2r_1 r_2 \cos(\theta_1 + \theta_2) + r_1^2 r_2^2} \end{aligned}$$

Hence, the gain of the closed loop is given by

$$\begin{aligned} |C(j\omega)| &= \frac{r_1 |e^{j\theta_1} + r_1 r_2 e^{-j\theta_2}|}{1 + 2r_1 r_2 \cos(\theta_1 + \theta_2) + r_1^2 r_2^2} \\ &= \frac{r_1 \sqrt{(e^{j\theta_1} + r_1 r_2 e^{-j\theta_2})(e^{-j\theta_1} + r_1 r_2 e^{j\theta_2})}}{1 + 2r_1 r_2 \cos(\theta_1 + \theta_2) + r_1^2 r_2^2} \\ &= \frac{r_1 \sqrt{1 + r_1 r_2 e^{j(\theta_1 + \theta_2)} + r_1 r_2 e^{-j(\theta_1 + \theta_2)} + r_1^2 r_2^2}}{1 + 2r_1 r_2 \cos(\theta_1 + \theta_2) + r_1^2 r_2^2} \\ &= \frac{r_1 \sqrt{1 + 2r_1 r_2 \cos(\theta_1 + \theta_2) + r_1^2 r_2^2}}{1 + 2r_1 r_2 \cos(\theta_1 + \theta_2) + r_1^2 r_2^2} \\ &= \frac{r_1}{\sqrt{1 + 2r_1 r_2 \cos(\theta_1 + \theta_2) + r_1^2 r_2^2}} \end{aligned}$$

and the phase shift is²

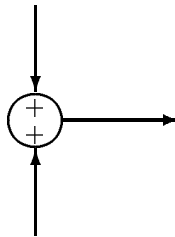
$$\begin{aligned} \arg(C(j\omega)) &= \arctan(\Im(C(j\omega))/\Re(C(j\omega))) \\ &= \arctan\left(\frac{\Im(e^{j\theta_1} + r_1 r_2 e^{-j\theta_2})}{\Re(e^{j\theta_1} + r_1 r_2 e^{-j\theta_2})}\right) \\ &= \arctan\left(\frac{\sin \theta_1 - r_1 r_2 \sin \theta_2}{\cos \theta_1 + r_1 r_2 \cos \theta_2}\right) \end{aligned}$$

²Some side conditions may be required on the use of arctan.

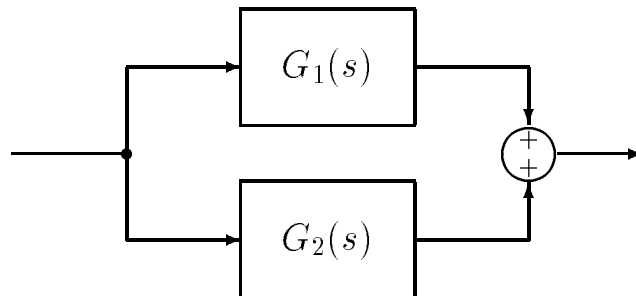
So, the gain and phase shift of a feedback loop can be expressed purely in terms of the gains and phase shifts of its subsystems, but in contrast to sequencing, in this case the gain and phase shifts become inter-dependent. As a consequence, in reasoning about frequency response, the gain and phase shift must be taken together.

3.3 Summing Points

Apart from feedback loops, the other significant structure in control systems is a summing point:



For single-input, single-output systems, it can be assumed that the two signals to be summed ultimately come from the same source:



The Laplace transform expressing the relationship between the output and input of this system is:

$$C(s) = G_1(s) + G_2(s)$$

where $G_1(s)$ and $G_2(s)$ are the transforms for the subsystems between the common input and the summing point. Once again expressing the complex numbers $G_1(j\omega)$ and $G_2(j\omega)$ as $r_1e^{j\theta_1}$ and $r_2e^{j\theta_2}$, respectively, the gain and phase shift of the full system in terms of the gains (r_1 and r_2) and phase shifts (θ_1 and θ_2) of the subsystems can be determined via the transform of the full system:

$$C(j\omega) = r_1e^{j\theta_1} + r_2e^{j\theta_2}$$

The gain is the modulus of $C(j\omega)$, which is equal to the square root of the product of $C(j\omega)$ and its conjugate:

$$\begin{aligned} |C(j\omega)| &= \sqrt{(r_1 e^{j\theta_1} + r_2 e^{j\theta_2})(r_1 e^{-j\theta_1} + r_2 e^{-j\theta_2})} \\ &= \sqrt{r_1^2 + r_1 r_2 e^{j(\theta_1 - \theta_2)} + r_1 r_2 e^{-j(\theta_1 - \theta_2)} + r_2^2} \\ &= \sqrt{r_1^2 + 2r_1 r_2 \cos(\theta_1 - \theta_2) + r_2^2} \end{aligned}$$

The phase shift is given by

$$\begin{aligned} \arg(C(j\omega)) &= \arctan(\Im(C(j\omega))/\Re(C(j\omega))) \\ &= \arctan\left(\frac{\Im(r_1 e^{j\theta_1} + r_2 e^{j\theta_2})}{\Re(r_1 e^{j\theta_1} + r_2 e^{j\theta_2})}\right) \\ &= \arctan\left(\frac{r_1 \sin \theta_1 + r_2 \sin \theta_2}{r_1 \cos \theta_1 + r_2 \cos \theta_2}\right) \end{aligned}$$

4 Cosy: A Simple Language for Control Systems

The sequencing, feedback loop, and summation constructs presented in Section 3 are sufficient, either directly or by means of equivalence-preserving transformations, to represent a wide range of control systems. All the constructs have a single input and a single output and are described in terms of subsystems that also have this property. Hence, they give rise to a simple language for control systems, which we shall call Cosy. The abstract syntax of this language is given in Figure 2, alongside the corresponding block diagrams.

In addition to the compound constructs, there are two atomic forms: **Unit**, which simply represents a wire, and **Fcn**, which represents a transfer function with gain dr and phase shift $d\theta$. The names dr and $d\theta$ are used instead of r and θ to emphasize that these values represent the *changes* in amplitude and phase caused by the transfer function. Expressions for dr and $d\theta$ can be obtained from the expression for the transfer function, so the latter could have been used in the **Fcn** form. In presenting an axiomatic semantics for Cosy, it is, however, simpler to have dr and $d\theta$ represented explicitly. The gain and phase shift of **Unit** are 1 and 0, respectively. C_1 and C_2 are arbitrary subsystems.

As far as we can see, loops cannot be represented in terms of sequencing and summation because of the feedback which introduces a cycle in the block diagram. (A block diagram can be viewed as a graph.) In any case,

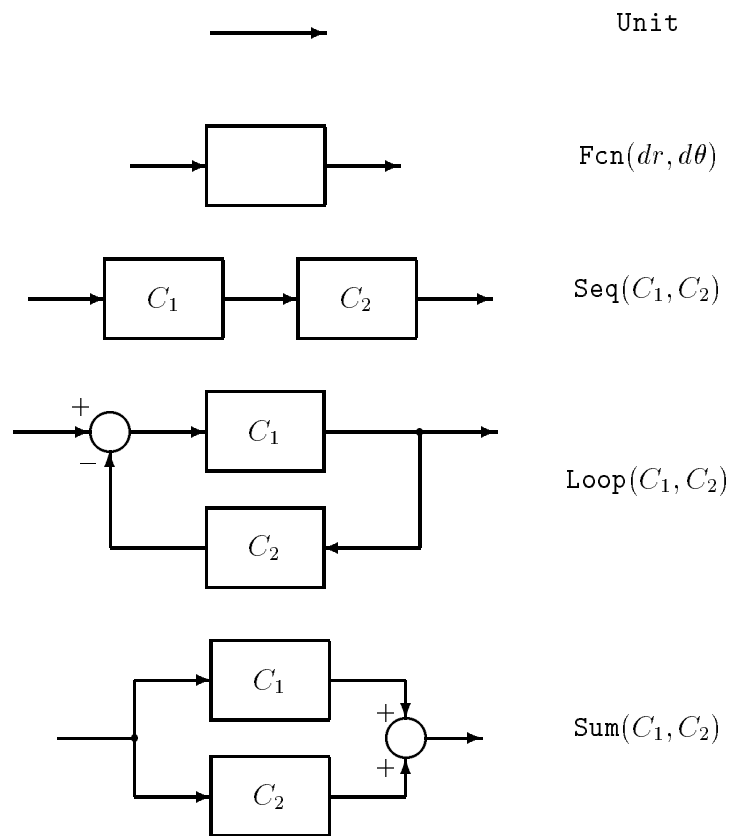
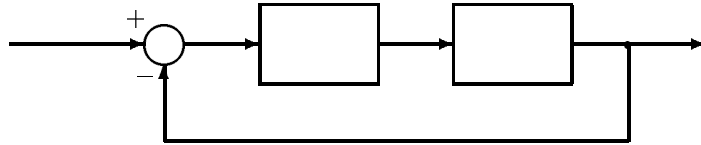


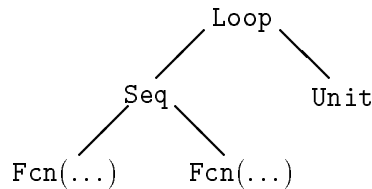
Figure 2: The Cosy language constructs

feedback loops are sufficiently common that they deserve to be a primitive structure of the language.

As an example, consider the following block diagram:



The abstract syntax tree for the Cosy representation of this system is:



A textual representation for the same is:

`Loop(Seq(Fcn(...), Fcn(...)), Unit)`

Observe how `Unit` is used for a subsystem that is just a wire. It is not strictly necessary to have `Unit` in the language — `Fcn(1,0)` could be used instead — but Cosy expressions are more elegant using `Unit`.

With this language, it is possible to write a Hoare-style logic for frequency response properties.

Thus far we have said nothing about which block diagrams can be represented in this way. Those which can will be called **tree-structured**. However a wider class can be represented due to the existence of block diagram equivalences: these essentially correspond to the block diagram representation of the distributive laws. In text-books [16] these are generally studied in an ad-hoc way: they can be given a more formal treatment through the study of the algebraic structure of feedback diagrams, and have intriguing links with similar equivalences in category theory [9]. There are examples of block diagrams are not equivalent to tree-structured diagrams, but the only ones we know of have a complicated nested structure which seems unlikely to occur in practice.

5 Hoare Rules for Cosy

The rules of a Hoare logic usually operate on triples consisting of a precondition (a predicate logic formula), a statement of the programming (or other) language, and a postcondition. In the Hoare logic for Cosy, an additional component is added: a pair consisting of the gain and phase shift induced by the control system represented by the Cosy expression.

More precisely, $\{P\}C\langle dr, d\theta\rangle\{Q\}$ 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. P and Q assert properties of gain and phase. In the original Hoare logic [10], the properties were for states involving a flexible number of program variables. For an analysis of frequency response of a control system, there are only two "variables" of interest: gain and phase. (Note, however, that these two variables are actually functions of a global universally quantified variable for frequency.)

In what follows, $P[r \setminus R, \theta \setminus \Theta]$ means the property P with references to the gain r replaced by the expression R , and references to the phase θ replaced by the expression Θ , e.g. $(r < 2)[r \setminus r * 3]$ means $r * 3 < 2$.

5.1 Definitions

The expressions for the gain and phase shift of loops and summations appear a lot within the Hoare rules, so it is worthwhile defining abbreviations for them:

$$\begin{aligned} lr(dr_1, d\theta_1, dr_2, d\theta_2) &= \frac{dr_1}{\sqrt{dr_1^2 dr_2^2 + 2dr_1 dr_2 \cos(d\theta_1 + d\theta_2) + 1}} \\ lt(dr_1, d\theta_1, dr_2, d\theta_2) &= \arctan\left(\frac{\sin d\theta_1 - dr_1 dr_2 \sin d\theta_2}{\cos d\theta_1 + dr_1 dr_2 \cos d\theta_2}\right) \\ sr(dr_1, d\theta_1, dr_2, d\theta_2) &= \sqrt{dr_1^2 + 2dr_1 dr_2 \cos(d\theta_1 - d\theta_2) + dr_2^2} \\ st(dr_1, d\theta_1, dr_2, d\theta_2) &= \arctan\left(\frac{dr_1 \sin d\theta_1 + dr_2 \sin d\theta_2}{dr_1 \cos d\theta_1 + dr_2 \cos d\theta_2}\right) \end{aligned}$$

As can be seen, these definitions use the results obtained in Section 3.

5.2 Axioms and Rules for the Primitive Forms

Presented below are an axiom or rule for each of the Cosy language constructs. For loops and summations, there are many variations the rules

could take. The rules given in this section are not the most general, but are arguably more natural than some of the other possibilities in that they make a connection between predicates appearing in the hypotheses based on the physical structure of the system. So, for example, the rule for loops has the same predicate Q for both the postcondition of C_1 and the precondition of C_2 , reflecting the physical connection between the output of C_1 and the input of C_2 .

The Unit Axiom

Unit has no effect on the signal, so the postcondition is the same as the precondition. The gain is 1, and the phase shift is 0.

$$\overline{\vdash \{P\}\mathbf{Unit}\langle 1, 0 \rangle \{P\}}$$

The (Transfer) Function Axiom

The axiom for transfer functions is analogous to the Assignment Axiom for imperative programming languages. If P is true at the input having had the variables r and θ (gain and phase) replaced by the values they have at the output, then P is true at the output. The value of r at the output is equal to the product of the value of r at the input with the function's gain dr . Similarly, the value of θ at the output is the sum of θ at the input and the function's phase shift $d\theta$.

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

The Sequencing Rule

The sequencing rule simply states that gain and phase compose as shown in Section 3. Note the connection between the postcondition of C_1 and the precondition of C_2 .

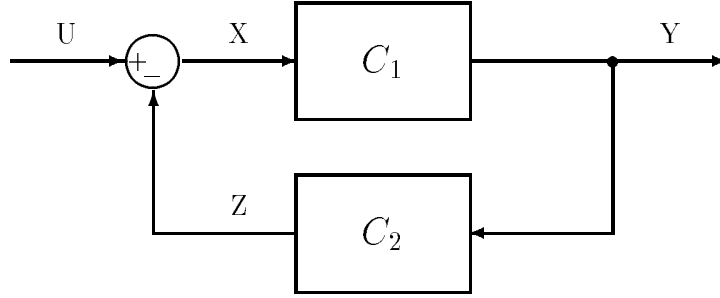
$$\frac{\vdash \{P\}C_1\langle dr_1, d\theta_1 \rangle \{Q\} \quad \vdash \{Q\}C_2\langle dr_2, d\theta_2 \rangle \{R\}}{\vdash \{P\}\mathbf{Seq}(C_1, C_2)\langle dr_1 * dr_2, d\theta_1 + d\theta_2 \rangle \{R\}}$$

A Loop Rule

The following rule for feedback loops reflects the physical connection between the two subsystems, through the common predicate Q .

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

Accepting that the aim in this rule is to have some instance of P as the precondition of the conclusion, and some instance of R as the postcondition, the actual substitutions required can be derived by reference to the following diagram:



Using the diagram, together with the hypotheses of the rule and the composition properties of gain and phase, the following reasoning can be made, where $P @ X$ means the predicate P is true at point X in the diagram:

$$\begin{aligned} R @ Y &\Leftrightarrow R[r \setminus r/dr_2, \theta \setminus \theta - d\theta_2] @ Z \\ &\Leftrightarrow Q[r \setminus r/dr_2, \theta \setminus \theta - d\theta_2] @ Y \\ &\Leftrightarrow P[r \setminus r/dr_2, \theta \setminus \theta - d\theta_2] @ X \\ &\Leftrightarrow (P[r \setminus r/dr_2, \theta \setminus \theta - d\theta_2])[r \setminus r/dr_1, \theta \setminus \theta - d\theta_1] @ Y \\ &= P[r \setminus r/(dr_1 * dr_2), \theta \setminus \theta - (d\theta_1 + d\theta_2)] @ Y \\ &\Leftrightarrow P[r \setminus r * lr(dr_1, d\theta_1, dr_2, d\theta_2)/(dr_1 * dr_2), \\ &\quad \theta \setminus \theta + lt(dr_1, d\theta_1, dr_2, d\theta_2) - (d\theta_1 + d\theta_2)] @ U \end{aligned}$$

The aim is to know something about R at the output of the loop, i.e. at Y. The reasoning proceeds by first moving through C_2 to Z using the transfer function axiom. Then the two hypotheses are used to move backwards in the signal flow to point X. That is done to obtain a predicate in terms of P .

The gain/phase relationship between point U (the input to the loop) and point X has not been worked out. So, it is easier to move forwards through C_2 and then use the gain/phase relationship between U and Y that has been worked out.

Finally, in order to avoid division and subtraction appearing in the rule, some of the gain/phase changes in the precondition are shifted into the postcondition. A general rule for this maneuver is presented below.

A Sum Rule

The following rule for summations can be derived in a similar way to the loop rule above, but here the two hypotheses share the same precondition:

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

5.3 Logical Rules

The Hoare logic for Cosy features the traditional Hoare rules for strengthening a precondition and weakening a postcondition, but it also has two other logical rules. Arbitrary amounts of gain and phase shift can be moved between the precondition and postcondition and vice versa. There are two rules for this, one for each direction. The only restriction is that the gain factor moved between the two conditions must be non-zero to avoid division by zero problems.

Precondition Strengthening

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

Postcondition Weakening

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

The Shift Right Rule

$$\frac{\vdash r' \neq 0 \quad \vdash \{P[r \setminus r * r', \theta \setminus \theta + \theta']\}C <dr, d\theta> \{Q\}}{\vdash \{P\}C <dr, d\theta> \{Q[r \setminus r/r', \theta \setminus \theta - \theta']\}}$$

The Shift Left Rule

$$\frac{\vdash r' \neq 0 \quad \vdash \{P\}C <dr, d\theta> \{Q[r \setminus r * r', \theta \setminus \theta + \theta']\}}{\vdash \{P[r \setminus r/r', \theta \setminus \theta - \theta']\}C <dr, d\theta> \{Q\}}$$

5.4 Other Rules

The Component Rule

The transfer function axiom can be generalised to cover any system. If the gain and phase shift of the system are known, the system can be treated in the same way as an atomic transfer function. The following rule achieves that:

$$\frac{\vdash \{Q\}C <dr, d\theta> \{R\}}{\vdash \{P[r \setminus r * dr, \theta \setminus \theta + d\theta]\}C <dr, d\theta> \{P\}}$$

It is a rule rather than an axiom (i.e. it has a hypothesis) because the gain and phase shift must be deduced somehow. Observe that the precondition Q and postcondition R of the hypothesis are completely arbitrary and independent of each other and of the precondition and postcondition of the conclusion. Only the values of dr and $d\theta$ are required from the hypothesis; the precondition and postcondition are thrown away.

A Generalised Loop Rule

The following rule is more general than the loop rule given above. It makes no connection between the postcondition of C_1 and the precondition of C_2 . Though arguably less intuitive, it is the version required to mechanise the logic in the form of a verification condition generator. As before, the substitutions in the precondition and postcondition of the conclusion have been normalised to avoid division and subtraction. In this case, it also allows the same substitution to be used for both P and R .

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

A Generalised Sum Rule

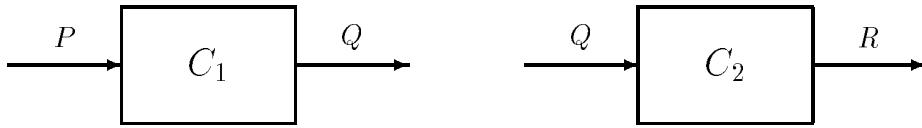
$$\frac{\vdash \{P\}C_1 \langle dr_1, d\theta_1 \rangle \{Q\} \quad \vdash \{R\}C_2 \langle dr_2, d\theta_2 \rangle \{S\}}{\vdash \{(P \wedge R)[r \setminus r * sr(dr_1, d\theta_1, dr_2, d\theta_2), \theta \setminus \theta + st(dr_1, d\theta_1, dr_2, d\theta_2)]\} \\ \text{Sum}(C_1, C_2) \langle sr(dr_1, d\theta_1, dr_2, d\theta_2), st(dr_1, d\theta_1, dr_2, d\theta_2) \rangle \\ \{Q[r \setminus r * dr_1, \theta \setminus \theta + d\theta_1] \wedge S[r \setminus r * dr_2, \theta \setminus \theta + d\theta_2]\}}$$

5.5 Justification for the Form of the Rules

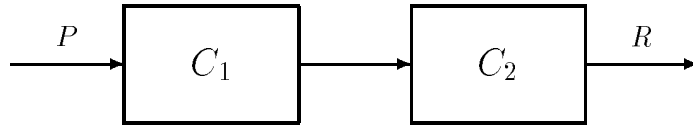
It is worth commenting on the addition of the gain/phase pair to the conventional Hoare triple. Ideally, the rules of the Hoare logic would manipulate preconditions and postconditions without any need to consider the phase and gain values explicitly, except when dealing with an atomic block. In a Hoare logic for an imperative programming language, only the rule (actually an axiom) for assignment statements explicitly mentions the values of the program variables. The rules for compound forms do not make such explicit reference. We have not been able to find such rules for Cosy, at least not without abstracting so much as to render the logic impotent.

In an imperative programming language, everything boils down to a sequence of assignment statements. If-then-else statements evaluate to either the then-branch or the else-branch, and while-loops are an iterated sequence of the body of the loop. In Cosy, there is something more going on. Feedback loops and summations are not equivalent to some sequence of the constituent blocks. Rather, their behaviour is determined by the way in which sine waves combine when superimposed.

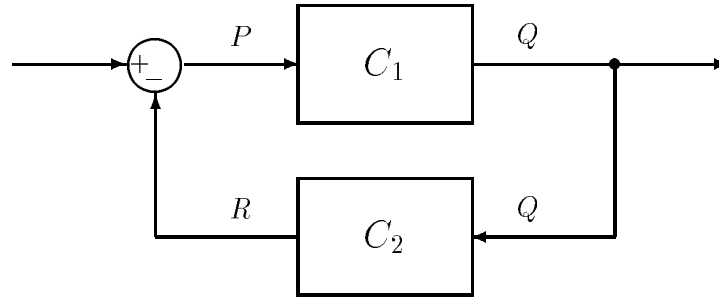
Consider the systems C_1 and C_2 in isolation, and suppose the predicates P , Q , and R (properties of gain and phase) hold at the points indicated below:



Because Q holds at the output of C_1 and also at the input of C_2 , it can be deduced that P and R hold as precondition and postcondition on the sequence of the two components:



This works because of the simple way in which gain and phase behave for sequencing. The same approach cannot be taken for a loop. The summing point prevents properties of the subsystems in isolation from being used to deduce properties for the loop without taking into account the way the summing point affects the gain and phase:



In the above, Q is the postcondition of the loop, but what is the precondition? How is it related to P and R ? The relationship is determined by the superposition of sine waves. That is the motivation for explicitly including the gain and phase in the rules of the Hoare logic. The question, then, is how best to include them.

At one extreme, the predicates could be omitted so that the rules manipulate only the gain and phase. This would be equivalent to only being able to talk about properties of the form $r = \dots \wedge \theta = \dots$, which would not be very interesting. At the least, inequalities on r and θ should be allowed.

A less extreme approach would be to include predicates, but not permit the property to change other than to reflect the changing values of gain and phase. This raises the question of whether the property has to be expressed explicitly in the rules. Could it not be treated as a global entity outside of the explicit representation of the proof? Indeed, one can argue that the logic would be no more than a way to compute the gain and phase, which would then be tested to see if they have the required property. A Hoare logic is not needed to achieve that.

No, it should be possible for the property to change within the proof, in addition to the gain and phase changing, e.g. by means of the precondition strengthening and postcondition weakening rules. This offers more flexible reasoning, e.g. having proved properties of two large subsystems, it is not necessary to have an exact match between the postcondition of one and the precondition of the other in order to establish a property of the whole; a logical implication is sufficient.

Having decided that gain and phase should be included explicitly in the logic, and that the preconditions and postconditions should be able to vary in ways other than those induced by the changes in the gain and phase, there is another choice. The gain and phase could be computed using a separate system from the main logic, to be introduced only as required. They would not then have to appear in the Hoare ‘triples’. Instead, some of the Hoare rules would have extra hypotheses asserting values of gain and phase for the subsystems, e.g.:

$$\frac{\begin{array}{l} \vdash \{P\}C_1\{Q\} \quad \vdash \text{GainPhase}(C_1) = (dr_1, d\theta_1) \\ \vdash \{Q\}C_2\{R\} \quad \vdash \text{GainPhase}(C_2) = (dr_2, d\theta_2) \end{array}}{\begin{array}{l} \vdash \{P[r \setminus r * lr(dr_1, d\theta_1, dr_2, d\theta_2), \theta \setminus \theta + lt(dr_1, d\theta_1, dr_2, d\theta_2)]\} \\ \text{Loop}(C_1, C_2) \\ \{R[r \setminus r * (dr_1 * dr_2), \theta \setminus \theta + (d\theta_1 + d\theta_2)]\} \end{array}}$$

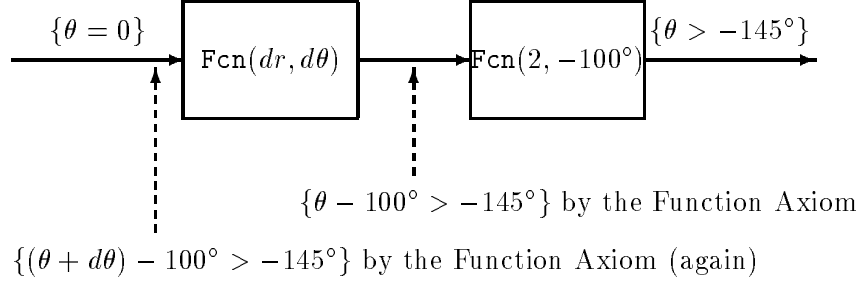
The gain and phase could be deduced using a separate set of rules, or, since the derivation is entirely deterministic, by some more algorithmic means. The argument against this separation is that it is likely to involve a lot of redundant reasoning/computation because obtaining values for a compound component involves obtaining values for the subcomponents. The latter will also have to be calculated when Hoare rules are applied to the subcomponents. Of course, there are ways around this duplication of effort, but doing the gain/phase calculation in the main rules of the logic seems to be a cleaner approach.

5.6 Example Proof

As a very simple example, suppose there are two components in sequence, a controller and a plant, and the phase shift of the combined system is required to be greater than -145° , as follows:

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

The gain and phase shift of the controller have been left as symbolic entities dr and $d\theta$. The aim is to determine constraints on them using the Hoare logic. The proof, using the function axiom and sequencing rule, can be illustrated like this:



More formally, the first step is to establish theorem **Th1** by the function axiom applied to the second component:

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

Then, taking the precondition of **Th1** as the postcondition for the first component, the function axiom gives:

$$\mathbf{Th2}: \vdash \{(\theta + d\theta) - 100^\circ > -145^\circ\} \mathbf{Fcn}(dr, d\theta) < dr, d\theta > \{\theta - 100^\circ > -145^\circ\}$$

Applying the sequencing rule to **Th1** and **Th2** yields:

$$\mathbf{Th3}: \vdash \{(\theta + d\theta) - 100^\circ > -145^\circ\} \\ \text{Seq}(\mathbf{Fcn}(dr, d\theta), \mathbf{Fcn}(2, -100^\circ)) < dr * 2, d\theta - 100^\circ > \\ \{\theta > -145^\circ\}$$

Now, if it can be proved that $(\theta = 0) \Rightarrow ((\theta + d\theta) - 100^\circ > -145^\circ)$, precondition strengthening can be used to get:

$$\mathbf{Th4}: \vdash \{\theta = 0\} \\ \text{Seq}(\mathbf{Fcn}(dr, d\theta), \mathbf{Fcn}(2, -100^\circ)) < dr * 2, d\theta - 100^\circ > \\ \{\theta > -145^\circ\}$$

Th4 is only a theorem under the above condition on θ . The condition is true if and only if $d\theta > -45^\circ$. This is a constraint on the controller.

In this example, the constraint on $d\theta$ could have been determined from the compound gain and phase shift $< dr * 2, d\theta - 100^\circ >$, but for large examples, the expressions for the overall gain and phase shift may become too complex to be practical. In contrast, the precondition and postcondition can be simpler, because they do not have to state equalities for r and θ , but need only place bounds on them.

6 Tactics and Verification Conditions

The Hoare rules are usable for small proofs, but for proofs of any size, reasoning forwards (i.e. bottom-up) is painful. Nor do the rules lend themselves directly to automation of proof. The usual approach to automating Hoare logics is to produce a *verification condition generator*. Verification conditions are the logical formulas that must ultimately be proved in a proof within the Hoare logic. The verification conditions themselves are pure predicate logic formulas, that is they do not involve the constructs of the language (in this case Cosy).

Following Gordon's approach to mechanising Hoare logic [7], we have produced *tactics* for the Hoare logic for Cosy. Tactics are procedures in a mechanised theorem proving system that implement *backward* reasoning. In a sense, tactics invert the inference rules. They allow one to start with the formula to be proved, and decompose it into simpler formulas. The decomposition continues until the formulas produced can be proved directly. Programs in the theorem proving system manage the decomposition process and the construction of a proof for the original formula from the proofs of the simpler formulas. Verification conditions are generated by recursive application of the tactics.

6.1 The Tactics

The tactics for Cosy are presented below. A tactic for summations would be similar to the tactic for loops, but we have not yet worked out the details. The notation for tactics consists of a Hoare-logic formula above a double line, and one or more Hoare-logic or predicate-logic formulas below the line. In applying the tactic, the formula above the line is matched against the formula to be proved and the resulting substitution is applied to the formulas below the line to produce the new formulas.

In some circumstances there is insufficient information in the formula to determine how to decompose it. In these situations, the formula must be annotated with additional information. To this end, two constructs are added to the Cosy language: **Assert** and **GainPhase**. **Assert** asserts that some property (predicate) holds at a point in the control system, while **GainPhase** asserts values for the gain and phase shift of a component. These two new constructs can be added to a system without changing its behaviour with respect to gain and phase. They exist only as a means of annotation for use by the verification condition generator.

Where division occurs in the tactics, it may be necessary to add other

formulas below the double line stating that the divisors are non-zero. Such formulas have been omitted in the presentation below, but see Section 7 for a discussion of this in the context of mechanisation.

The Unit Tactic

The tactic for the `Unit` construct is straightforward. Variables dr and $d\theta$ are used in the formula above the line so that it matches any Hoare-logic formula for `Unit`. The new formulas to be proved say that the instantiations of dr and $d\theta$ must be equal to 1 and 0, respectively. The precondition and postcondition above the line are also as general as they can be. It must be shown that P logically implies Q so that precondition strengthening can be used along with the unit axiom to justify the original formula.

$$\frac{\{P\}\text{Unit}\langle dr, d\theta \rangle\{Q\}}{dr = 1 \quad d\theta = 0 \quad P \Rightarrow Q}$$

The (Transfer) Function Tactic

The tactic for transfer functions is analogous to the tactic for `Unit`, only a bit more general.

$$\frac{\{P\}\text{Fcn}(dr, d\theta)\langle dr', d\theta' \rangle\{Q\}}{dr = dr' \quad d\theta = d\theta' \quad P \Rightarrow Q[r \setminus r * dr, \theta \setminus \theta + d\theta]}$$

The Sequencing Tactic (multiple cases)

The tactic for sequencing consists of several cases. When one or both of the components is an atomic transfer function, there is no need for the formula to be annotated. One transfer function provides enough information for the tactic to know how to decompose the formula. However, when neither component is a transfer function, annotations must be present. There is no need to have cases in which one of the components is a `Unit`, because a sequence involving a `Unit` can be simplified, e.g. `Seq(Unit, C)` can be simplified to `C`. The tactic also assumes that the formula has been normalised so that `Asserts` are associated with the second component.

$$\frac{\{P\}\text{Seq}(C_1, \text{Fcn}(dr_2, d\theta_2))\langle dr, d\theta \rangle\{Q\}}{\{P\}C_1\langle dr/dr_2, d\theta - d\theta_2 \rangle\{Q[r \setminus r * dr_2, \theta \setminus \theta + d\theta_2]\}}$$

$$\frac{\{P\}\text{Seq}(\text{Fcn}(dr_1, d\theta_1), C_2) \langle dr, d\theta \rangle \{Q\}}{\{P[r \setminus r/dr_1, \theta \setminus \theta - d\theta_1]\}C_2 \langle dr/dr_1, d\theta - d\theta_1 \rangle \{Q\}}$$

$$\frac{\{P\}\text{Seq}(C_1, \text{Seq}(\text{Assert}(R), \text{GainPhase}(C_2, (dr_2, d\theta_2)))) \langle dr, d\theta \rangle \{Q\}}{\{P\}C_1 \langle dr/dr_2, d\theta - d\theta_2 \rangle \{R\} \quad \{R\}C_2 \langle dr_2, d\theta_2 \rangle \{Q\}}$$

$$\frac{\{P\}\text{Seq}(\text{GainPhase}(C_1, (dr_1, d\theta_1)), \text{Seq}(\text{Assert}(R), C_2)) \langle dr, d\theta \rangle \{Q\}}{\{P\}C_1 \langle dr_1, d\theta_1 \rangle \{R\} \quad \{R\}C_2 \langle dr/dr_1, d\theta - d\theta_1 \rangle \{Q\}}$$

The Loop Tactic (multiple cases)

Like the tactic for sequences, the tactic for loops has multiple cases. For loops, the cases where one of the components is **Unit** have to be considered explicitly because the Cosy expression does not reduce. No **Asserts** are required, but the cases for two non-atomic subcomponents have to be annotated with a **GainPhase** construct. In those cases, two new formulas are created by application of the tactic. The second formula is required only to prove gain and phase shift values for C_2 , so the precondition and postcondition are each the trivial predicate **true**.

$$\frac{\{P\}\text{Loop}(C_1, \text{Unit}) \langle dr, d\theta \rangle \{Q\}}{\{P[r \setminus r/dr, \theta \setminus \theta - d\theta]\} \\ C_1 \langle lr(dr, d\theta, -1, 0), lt(dr, d\theta, -1, 0) \rangle \\ \{Q[r \setminus r/lr(dr, d\theta, -1, 0), \theta \setminus \theta - lt(dr, d\theta, -1, 0)]\}}$$

$$\frac{\{P\}\text{Loop}(C_1, \text{Fcn}(dr_2, d\theta_2)) \langle dr, d\theta \rangle \{Q\}}{\{P[r \setminus r/dr, \theta \setminus \theta - d\theta]\} \\ C_1 \langle lr(dr, d\theta, -dr_2, d\theta_2), lt(dr, d\theta, -dr_2, d\theta_2) \rangle \\ \{Q[r \setminus r/lr(dr, d\theta, -dr_2, d\theta_2), \theta \setminus \theta - lt(dr, d\theta, -dr_2, d\theta_2)]\}}$$

$$\frac{\{P\}\text{Loop}(\text{Unit}, C_2) \langle dr, d\theta \rangle \{Q\}}{\{P[r \setminus r/dr, \theta \setminus \theta - d\theta]\} \\ C_2 \langle 1/lr(dr, d\theta, -1, 0), -lt(dr, d\theta, -1, 0) \rangle \\ \{Q[r \setminus r * lr(dr, d\theta, -1, 0), \theta \setminus \theta + lt(dr, d\theta, -1, 0)]\}}$$

$$\frac{\{P\}\text{Loop}(\text{Fcn}(dr_1, d\theta_1), C_2) < dr, d\theta > \{Q\}}{\{P[r \setminus r/dr, \theta \setminus \theta - d\theta]\}} \\ C_2 < 1/lr(dr, d\theta, -1/dr_1, -d\theta_1), -lt(dr, d\theta, -1/dr_1, -d\theta_1) > \\ \{Q[r \setminus r * lr(dr, d\theta, -1/dr_1, -d\theta_1), \theta \setminus \theta + lt(dr, d\theta, -1/dr_1, -d\theta_1)]\}$$

$$\frac{\{P\}\text{Loop}(C_1, \text{GainPhase}(C_2, (dr_2, d\theta_2))) < dr, d\theta > \{Q\}}{\{P[r \setminus r/dr, \theta \setminus \theta - d\theta]\}} \\ C_1 < lr(dr, d\theta, -dr_2, d\theta_2), lt(dr, d\theta, -dr_2, d\theta_2) > \\ \{Q[r \setminus r/lr(dr, d\theta, -dr_2, d\theta_2), \theta \setminus \theta - lt(dr, d\theta, -dr_2, d\theta_2)]\} \wedge \\ \{\text{true}\}C_2 < dr_2, d\theta_2 > \{\text{true}\}$$

$$\frac{\{P\}\text{Loop}(\text{GainPhase}(C_1, (dr_1, d\theta_1)), C_2) < dr, d\theta > \{Q\}}{\{P[r \setminus r/dr, \theta \setminus \theta - d\theta]\}C_1 < dr_1, d\theta_1 > \{Q[r \setminus r/dr_1, \theta \setminus \theta - d\theta_1]\} \wedge \\ \{\text{true}\}C_2 < 1/lr(dr, d\theta, -1/dr_1, -d\theta_1), -lt(dr, d\theta, -1/dr_1, -d\theta_1) > \{\text{true}\}}$$

6.2 Derivation of Gain and Phase Expressions for the Loop Tactic

The expressions for the gain and phase shift in the loop tactic are derived by the same process as used in Section 3.2. Here, the aim is to find expressions for one of the subcomponents in terms of the gains and phase shifts of the loop and the other subcomponent, e.g. r_1 in terms of r , θ , r_2 , and θ_2 .

Recall that

$$C(s) = \frac{G_1(s)}{1 + G_1(s)G_2(s)}$$

Therefore,

$$G_1(s) = \frac{C(s)}{1 - C(s)G_2(s)} = \frac{C(s)}{1 + C(s)(-G_2(s))}$$

This expression for $G_1(s)$ has the same shape as the earlier expression for $C(s)$. So, by analogy,

$$|G_1(j\omega)| = lr(r, \theta, -r_2, \theta_2)$$

and

$$\arg(G_1(j\omega)) = lt(r, \theta, -r_2, \theta_2)$$

Unfortunately, obtaining expressions for r_2 and θ_2 in terms of r , θ , r_1 , and θ_1 is not so simple. Rearranging the equation for $C(s)$ again, this time to isolate $G_2(s)$, gives:

$$G_2(s) = \frac{1}{C(s)} - \frac{1}{G_1(s)}$$

It is better, however, to derive an expression for the inverse of $G_2(s)$ because that yields the familiar shape:

$$\frac{1}{G_2(s)} = \frac{G_1(s)C(s)}{G_1(s) - C(s)} = \frac{C(s)}{1 - \frac{C(s)}{G_1(s)}} = \frac{C(s)}{1 + C(s)\frac{1}{-G_1(s)}}$$

Now,

$$\frac{1}{-G_1(j\omega)} = \frac{1}{-r_1 e^{j\theta_1}} = \frac{-e^{-j\theta_1}}{r_1} = \left(\frac{-1}{r_1}\right) e^{j(-\theta_1)}$$

Hence,

$$\left| \frac{1}{G_2(j\omega)} \right| = lr(r, \theta, -1/r_1, -\theta_1)$$

and

$$\arg\left(\frac{1}{G_2(j\omega)}\right) = lt(r, \theta, -1/r_1, -\theta_1)$$

But, for a complex number z , $|1/z| = 1/|z|$, and $\arg(1/z) = -\arg(z)$. So,

$$|G_2(j\omega)| = \frac{1}{lr(r, \theta, -1/r_1, -\theta_1)}$$

and

$$\arg(G_2(j\omega)) = -lt(r, \theta, -1/r_1, -\theta_1)$$

6.3 Example Justification Proof

Now consider as an example the second case of the loop tactic. A theorem for the formula above the lines can be obtained from a theorem for the formula below the lines as follows. The initial theorem is:

$$\begin{aligned} \vdash \{ & P[r \setminus r/dr, \theta \setminus \theta - d\theta] \\ & C_1 < lr(dr, d\theta, -dr_2, d\theta_2), lt(dr, d\theta, -dr_2, d\theta_2) > \\ & \{Q[r \setminus r/lr(dr, d\theta, -dr_2, d\theta_2), \theta \setminus \theta - lt(dr, d\theta, -dr_2, d\theta_2)]\} \} \end{aligned} \quad (1)$$

Another theorem can be obtained by using the transfer function axiom on $\text{Fcn}(dr_2, d\theta_2)$ with true as the postcondition:

$$\vdash \{\text{true}[r \setminus r * dr_2, \theta \setminus \theta + d\theta_2]\}\text{Fcn}(dr_2, d\theta_2) \langle dr_2, d\theta_2 \rangle \{\text{true}\}$$

This simplifies to:

$$\vdash \{\text{true}\}\text{Fcn}(dr_2, d\theta_2) \langle dr_2, d\theta_2 \rangle \{\text{true}\} \quad (2)$$

Applying the generalised loop rule to (1) and (2) gives:

$$\begin{aligned} \vdash \{ & (P[r \setminus r/dr, \theta \setminus \theta - d\theta] \wedge \text{true}) \\ & [r \setminus r * lr(lr(dr, d\theta, -dr_2, d\theta_2), lt(dr, d\theta, -dr_2, d\theta_2), dr_2, d\theta_2), \\ & \theta \setminus \theta + lt(lr(dr, d\theta, -dr_2, d\theta_2), lt(dr, d\theta, -dr_2, d\theta_2), dr_2, d\theta_2)]\} \\ & \text{Loop}(C_1, \text{Fcn}(dr_2, d\theta_2)) \langle lr(lr(dr, d\theta, -dr_2, d\theta_2), lt(dr, d\theta, -dr_2, d\theta_2), dr_2, d\theta_2), \\ & \quad lt(lr(dr, d\theta, -dr_2, d\theta_2), lt(dr, d\theta, -dr_2, d\theta_2), dr_2, d\theta_2) \rangle > \\ & \{(Q[r \setminus r/lr(dr, d\theta, -dr_2, d\theta_2), \theta \setminus \theta - lt(dr, d\theta, -dr_2, d\theta_2)] \\ & \quad [r \setminus r * lr(dr, d\theta, -dr_2, d\theta_2), \theta \setminus \theta + lt(dr, d\theta, -dr_2, d\theta_2)] \wedge \\ & \quad \text{true}[r \setminus r * dr_2, \theta \setminus \theta + d\theta_2])\} \end{aligned} \quad (3)$$

Now,

$$lr(lr(dr, d\theta, -dr_2, d\theta_2), lt(dr, d\theta, -dr_2, d\theta_2), dr_2, d\theta_2) = dr \quad (4)$$

$$lt(lr(dr, d\theta, -dr_2, d\theta_2), lt(dr, d\theta, -dr_2, d\theta_2), dr_2, d\theta_2) = d\theta \quad (5)$$

So, (3) simplifies to:

$$\begin{aligned} \vdash \{ & (P[r \setminus r/dr, \theta \setminus \theta - d\theta])[r \setminus r * dr, \theta \setminus \theta + d\theta] \\ & \text{Loop}(C_1, \text{Fcn}(dr_2, d\theta_2)) \langle dr, d\theta \rangle \\ & \{(Q[r \setminus r/lr(dr, d\theta, -dr_2, d\theta_2), \theta \setminus \theta - lt(dr, d\theta, -dr_2, d\theta_2)] \\ & \quad [r \setminus r * lr(dr, d\theta, -dr_2, d\theta_2), \theta \setminus \theta + lt(dr, d\theta, -dr_2, d\theta_2)])\} \end{aligned}$$

Combining the substitutions on the predicates yields:

$$\begin{aligned} \vdash \{ & P[r \setminus (r * dr)/dr, \theta \setminus (\theta + d\theta) - d\theta] \\ & \text{Loop}(C_1, \text{Fcn}(dr_2, d\theta_2)) \langle dr, d\theta \rangle \\ & \{Q[r \setminus (r * lr(dr, d\theta, -dr_2, d\theta_2))/lr(dr, d\theta, -dr_2, d\theta_2), \\ & \quad \theta \setminus (\theta + lt(dr, d\theta, -dr_2, d\theta_2)) - lt(dr, d\theta, -dr_2, d\theta_2)]\} \end{aligned}$$

And, finally, some simple arithmetic makes the substitutions trivial, giving a theorem for the formula above the line in the tactic:

$$\vdash \{P\}\text{Loop}(C_1, \text{Fcn}(dr_2, d\theta_2)) \langle dr, d\theta \rangle \{Q\}$$

This proof constitutes a formal justification of the second case of the loop tactic. The other cases are similar given the following equations:

$$lr(dr_1, d\theta_1, 1/lr(dr, d\theta, -1/dr_1, -d\theta_1), -lt(dr, d\theta, -1/dr_1, -d\theta_1)) = d(6)$$

$$lt(dr_1, d\theta_1, 1/lr(dr, d\theta, -1/dr_1, -d\theta_1), -lt(dr, d\theta, -1/dr_1, -d\theta_1)) = d(7)$$

Equations (4), (5), (6), and (7) follow from the previously derived relationships between dr_1 , $d\theta_1$, dr_2 , $d\theta_2$, dr , and $d\theta$.

7 Mechanisation in HOL

The Hoare logic for Cosy has been mechanised in a version of the HOL system, specifically the Taupo-5 version of HOL98. HOL is similar to the ProofPower tool used in the ClawZ work described in the introduction [3], and the mechanisation could readily be adapted to work with ProofPower, provided the latter had all the necessary infrastructure such as a theory of transcendental functions.

A recursive type representing the abstract syntax of the Cosy language has been defined in the logic of the HOL system, and the behaviour of gain and phase defined over that type. The type includes constructors for the assertions `Assert` and `GainPhase`. (`Assert` has the same gain and phase shift as `Unit`.) The functions `lr`, `lt`, `sr`, and `st` are defined using square root and trigonometric functions from HOL's theories of the real numbers and transcendental functions. It is then simply assumed that `lr`, etc., are the correct expressions for the gain and phase shift of loops and summations. A more thorough formalisation would derive them from a theory of Laplace transforms.

Let $\llbracket C \rrbracket$ denote the gain and phase shift of C (as a pair). Then a statement of the Hoare logic is formalised using the constant `Spec`, defined as:

$$\text{Spec } P \ C \ (dr, d\theta) \ Q \ = \ (\llbracket C \rrbracket = (dr, d\theta)) \ \wedge \ (\forall r \ \theta. P(r, \theta) \Rightarrow Q(r * dr, \theta + d\theta))$$

In HOL, functions may take their arguments in a *curried* fashion, i.e., one at a time, written separated by spaces. Note also that the precondition and postcondition, P and Q , explicitly take a gain/phase pair as an argument. Substitutions on a predicate can then be written as a function composition between the predicate and a mapping from one gain/phase pair to another, e.g. $P[r \setminus r * dr, \theta \setminus \theta + d\theta]$ is formalised as:

$$P \circ (\lambda(r, \theta). (r * dr, \theta + d\theta))$$

The conjunction on predicates used in some of the rules and tactics can be defined by ‘lifting’ the usual logical ‘and’ to the level of predicates, i.e.:

$$\mathbf{AND} P Q = \lambda x. P(x) \wedge Q(x)$$

For our purposes, x in the above definition is a gain/phase pair, but the definition can be more generic.

The Hoare logic axioms and rules are proved as HOL theorems using the definitions of $\llbracket _ \rrbracket$ and \mathbf{Spec} . Some theorems of real arithmetic are also required in these proofs. The theorems for rules take the form of implications, e.g. the theorem for sequencing looks like this:

$$\mathbf{Spec} P C_1 (dr_1, d\theta_1) Q \wedge \mathbf{Spec} Q C_2 (dr_2, d\theta_2) R \Rightarrow \mathbf{Spec} P (\mathbf{Seq} C_1 C_2) (dr_1 * dr_2, d\theta_1 + d\theta_2) R$$

Having proved the axioms and rules as theorems in HOL, functions in the meta-language (a programming language) of HOL can be written to automate forward reasoning. The rules are then used in the justification proofs of tactics. The tactics are also meta-language functions and implement backward reasoning. A significant amount of expression simplification is required in the implementations of tactics due, amongst other things, to the representation of substitutions as function composition.

The presence of annotations complicates the implementation of tactics somewhat. The justification of each tactic uses the corresponding rule to construct a theorem for the original formula. For some of the tactics, however, the original formula had to include annotations. The result of applying the corresponding rule is an equivalent formula but without the annotations. For example, the target formula of one of the sequencing tactic cases is

$$\{P\}\mathbf{Seq}(C_1, \mathbf{Seq}(\mathbf{Assert}(R), \mathbf{GainPhase}(C_2, (dr_2, d\theta_2)))) \langle dr, d\theta \rangle \{Q\}$$

but the sequencing rule produces the following assertion-free formula:

$$\{P\}\mathbf{Seq}(C_1, C_2) \langle dr, d\theta \rangle \{Q\}$$

For a proper implementation of the tactic, a theorem for the formula involving annotations must be obtained from the theorem without annotations. The formulas are equivalent in the sense that the annotated and unannotated control systems have the same gain and phase shift. To implement the tactics, then, a special procedure has been written to prove the equivalence with respect to gain/phase. This uses theorems for congruence properties of the gain/phase function with respect to the constructs of *Cosy*.

If the Cosy constructs had been represented directly by their meanings as in Gordon's work [7], instead of as a recursive type over which the gain and phase are defined, the special procedure would have been unnecessary.

Another issue is that wherever a division appears in a tactic, the divisor must be shown to be non-zero. This is a consequence of the way in which real numbers are formalised in the HOL theories. In the implementation, these extra proof obligations are presented to the user along with any other formulas that could not be discharged automatically.

The individual cases of the sequencing tactic are combined into a single sequencing tactic called `Seq_TAC` by trying each case in turn until one matches. The same applies to the loop tactic. Some care is required in the order in which the cases are tried. The more general cases should be tried later to prevent them from being applied when a more specific case should be used.

Here is a fragment of a HOL session in which a formula of the Hoare logic is decomposed into simpler problems:

```
> val it =
  Proof manager status: 1 proof.
  1. Incomplete:
    Initial goal:
      Spec (\(r,th). r = 2) (Seq (Fcn (2,pi / 2)) Unit) (2,pi / 2)
        (\(r,th). r < 5)

    : proofs
- e (REPEAT (Seq_TAC ORELSE Fcn_TAC ORELSE Unit_TAC));
OK..
4 subgoals:
> val it =
  !r. (r / 2 = 2) ==> r < 5

  pi / 2 - pi / 2 = 0

  2 / 2 = 1

  ~(2 = 0)

  : goalstack
```

The ML function `REPEAT` repeatedly applies its argument tactic to the orig-

inal formula and any new formulas generated. **ORELSE** (an infix function) is used to try a number of tactics until one is found to be applicable. The residual formulas are the verification conditions (VCs). By including a linear arithmetic decision procedure in the compound tactic, two of the VCs can be proved automatically, so that only two remain:

```
- e (REPEAT (Seq_TAC ORELSE Fcn_TAC ORELSE Unit_TAC ORELSE
             RealArith.REAL_ARITH_TAC));
```

OK..

2 subgoals:

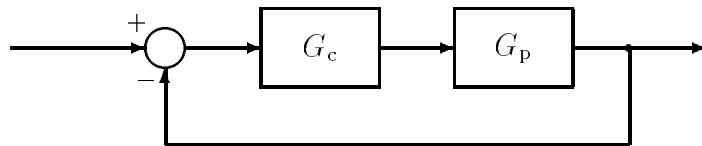
```
> val it =
  !r. (r / 2 = 2) ==> r < 5
```

```
2 / 2 = 1
```

The VCs remaining after applying the tactics often belong to the theory of real arithmetic including addition and multiplication (*elementary real algebra*). This theory is known to be decidable, so such verification conditions could, in theory, be discharged automatically. Harrison [8, Chapter 5] produced an implementation of a decision procedure for elementary real algebra in his version of HOL, but it is questionable whether it is efficient enough to be of use. It is worth noting, however, that this procedure was for the full theory with both universal and existential quantifiers. The verification conditions do not involve existential quantifiers, so a more efficient procedure may well be achievable. A state-of-the-art tool that could be used for deciding (full) elementary real algebra is QEPCAD. Tiwari at SRI International has linked this tool to the PVS theorem prover.³

8 A Larger Example

Consider the following closed-loop system:



³<http://www.csl.sri.com/users/tiwari/qepcad.html>

where the controller and plant are given by the following transfer functions:

$$G_c(s) = \frac{7s + 2}{s} \quad G_p(s) = \frac{1}{s^2 + 6s - 1.25}$$

The gains r_c and r_p , and the phase shifts θ_c and θ_p are given by the following expressions in terms of the frequency ω :

$$\begin{aligned} r_c = |G_c(j\omega)| &= \frac{\sqrt{49\omega^2 + 4}}{\omega} \\ \theta_c = \arg(G_c(j\omega)) &= \arctan\left(\frac{-2}{7\omega}\right) \\ r_p = |G_p(j\omega)| &= \frac{1}{\sqrt{\omega^4 + 38.5\omega^2 + 1.5625}} \\ \theta_p = \arg(G_p(j\omega)) &= \arctan\left(\frac{24\omega}{4\omega^2 + 5}\right) \end{aligned}$$

The aim is to prove the following Hoare logic formula:

$$\begin{aligned} &\{r = 1 \wedge \theta = 0\} \\ &\text{Seq}(\text{Fcn}(r_c, \theta_c), \text{Fcn}(r_p, \theta_p)) < dr, d\theta > \\ &\{(-215^\circ < \theta \wedge \theta < -145^\circ) \Rightarrow (r < \frac{1}{2} \vee r > 2)\} \end{aligned}$$

This formula describes the open-loop behaviour of the system. It asserts that within a region close to a -180° phase shift, the gain is either small or large relative to unit gain (0dB).⁴

Applying one of the cases of the sequencing tactic to the target formula yields

$$\begin{aligned} &\{r = 1 \wedge \theta = 0\} \\ &\text{Fcn}(r_c, \theta_c) < dr/r_p, d\theta - \theta_p > \\ &\{(-215^\circ < \theta + \theta_p \wedge \theta + \theta_p < -145^\circ) \Rightarrow (r * r_p < \frac{1}{2} \vee r * r_p > 2)\} \end{aligned}$$

together with the requirement that $r_p \neq 0$. Then, applying the transfer function tactic to the residual Hoare logic formula produces three formulas of pure predicate logic:

$$\begin{aligned} r_c &= dr/r_p \\ \theta_c &= d\theta - \theta_p \end{aligned}$$

$$\begin{aligned} &(r = 1 \wedge \theta = 0) \Rightarrow \\ &((-215^\circ < (\theta + \theta_c) + \theta_p \wedge (\theta + \theta_c) + \theta_p < -145^\circ) \Rightarrow ((r * r_c) * r_p < \frac{1}{2} \vee (r * r_c) * r_p > 2)) \end{aligned}$$

⁴This property corresponds to a naive rectangular exclusion region on a Nichols plot.

The first and second formulas can be satisfied by choosing appropriate values for dr and $d\theta$. For the third formula, the values of r and θ specified by the assumption can be substituted into the conclusion. The formula then simplifies to:

$$(-215^\circ < \theta_c + \theta_p \wedge \theta_c + \theta_p < -145^\circ) \Rightarrow (r_c * r_p < \frac{1}{2} \vee r_c * r_p > 2)$$

This, of course, is simply a constraint on the gain and phase shift of the result of composing G_c and G_p .

Now,

$$\begin{aligned} r_c * r_p &= \frac{\sqrt{49\omega^2 + 4}}{\omega\sqrt{\omega^4 + 38.5\omega^2 + 1.5625}} \\ \theta_c + \theta_p &= \arctan\left(\frac{-2}{7\omega}\right) + \arctan\left(\frac{24\omega}{4\omega^2 + 5}\right) \\ &= \arctan\left(\frac{\frac{-2}{7\omega} + \frac{24\omega}{4\omega^2 + 5}}{1 - \frac{-2}{7\omega} \frac{24\omega}{4\omega^2 + 5}}\right) \\ &= \arctan\left(\frac{-2(4\omega^2 + 5) + 168\omega^2}{7\omega(4\omega^2 + 5) + 48\omega}\right) \\ &= \arctan\left(\frac{160\omega^2 - 10}{28\omega^3 + 83\omega}\right) \end{aligned}$$

Also, in the range $(-215^\circ, -145^\circ)$, \tan is monotonically increasing, so it is sufficient to prove the following formula:

$$(\tan(-215^\circ) < \tan(\theta_c + \theta_p) \wedge \tan(\theta_c + \theta_p) < \tan(-145^\circ)) \Rightarrow (r_c * r_p < \frac{1}{2} \vee r_c * r_p > 2)$$

Using the fact that $\tan(-215^\circ)$ is slightly greater than $-\frac{71}{100}$, and $\tan(-145^\circ)$ is slightly less than $\frac{71}{100}$, it is also sufficient to prove:

$$\left(-\frac{71}{100} < \tan(\theta_c + \theta_p) \wedge \tan(\theta_c + \theta_p) < \frac{71}{100}\right) \Rightarrow (r_c * r_p < \frac{1}{2} \vee r_c * r_p > 2)$$

Substituting for $r_c * r_p$ and $\theta_c + \theta_p$ yields:

$$\left(-\frac{71}{100} < \frac{160\omega^2 - 10}{28\omega^3 + 83\omega} \wedge \frac{160\omega^2 - 10}{28\omega^3 + 83\omega} < \frac{71}{100}\right) \Rightarrow \left(\frac{\sqrt{49\omega^2 + 4}}{\omega\sqrt{\omega^4 + 38.5\omega^2 + 1.5625}} < \frac{1}{2} \vee \frac{\sqrt{49\omega^2 + 4}}{\omega\sqrt{\omega^4 + 38.5\omega^2 + 1.5625}} > 2\right)$$

With sufficient attention to side-conditions, this can be reduced to inequalities on polynomials in the frequency ω . These inequalities are constraints on ω in order for the original Hoare logic formula to hold. Assuming $\omega > 0$, the formula can be rewritten as:

$$\begin{aligned} (-71(28\omega^3 + 83\omega) < 100(160\omega^2 - 10) \wedge 100(160\omega^2 - 10) < 71(28\omega^3 + 83\omega)) \Rightarrow \\ (4(49\omega^2 + 4) < \omega^2(\omega^4 + 38.5\omega^2 + 1.5625) \vee 49\omega^2 + 4 > 4\omega^2(\omega^4 + 38.5\omega^2 + 1.5625)) \end{aligned}$$

This formula is within the scope of a decision procedure for elementary real algebra. In fact, it lies within the sub-theory in which there are no existentially quantified variables.

Proofs of properties of closed-loop systems follow a similar course, but are too complex to include in this paper. Any example involving the `Loop` construct is best handled in a mechanised theorem proving system.

9 Adapting to Discrete-Time Systems

This work has addressed control systems in the continuous-time domain. No substantive investigation of an analogous approach to discrete-time systems has been made, but it is possible to speculate on the similarities and differences.

The gain and phase shift of atomic blocks may be different in the discrete-time domain, even for analogous functions. However, it appears that the expressions for the composition of gain and phase for sequences, loops, and summations, are the same in the discrete-time domain as in the continuous-time domain. Matters are complicated when discrete and continuous blocks are mixed. If a continuous-time plant together with a preceding hold function are taken as a unit, i.e. as a single discrete-time block, the complications may be avoided.

Another difference from the continuous-time domain is that the theory breaks down for frequencies that are high (or low?) relative to the sampling frequency of the discrete-time system. This suggests that bounds will have to be placed on the range of frequencies for which a property is required to hold.

10 Conclusions and Future Work

This paper has proposed a framework for formally reasoning about the frequency response of continuous-time control systems. The framework takes the form of a Hoare-style logic, which is amenable to mechanisation in a theorem proving program. The logic is based around expressions for the gain and phase shift of various control-system structures in terms of the gain and phase shift of substructures.

The framework can be used to derive expressions for the gain and phase of a control system. This goes beyond usual control engineering practice because it can be used for symbolic values as readily as for numeric values. Furthermore, using a Hoare-style logic offers more than calculation (numeric

or symbolic): Constraints (typically inequalities) can be derived. This can be easier than seeking exact values for gain and phase. The approach can also be used in reverse to obtain constraints on the parameters of a controller so as to achieve specified behaviour of the overall system. The constraints indicate to the control engineer the size of the design space in which (s)he is working. Constraints may be derived for a particular frequency, a range of frequencies, or for all frequencies. Unlike many frequency response analysis methods, this approach is not limited to constraints that can be expressed graphically. And all this is done within the rigors of a formal proof system.

The logic allows a system to be treated as a hierarchy of subsystems. Properties of the full system may be proved from properties of the subsystems. This offers the potential to prove properties of commonly used components or subsystems just once. When a subsystem is re-used, it may not be necessary to do any further proof or derivation. Furthermore, the subsystem may be parameterised, which makes re-use more feasible.

The logic might also be used to prove that certain transformations on block diagrams preserve gain and phase properties.

One might argue that the full Hoare logic is unnecessary since the Component Rule can be used to establish a property for a full system without breaking it down into subcomponents (other than to compute the gain and phase expressions). However, with such an approach there would be one very complex verification condition, which would likely be more difficult to prove than the VCs generated by applying the other rules at the level of individual transfer functions. Using the Component Rule on the whole system also eliminates the option of using precondition strengthening and postcondition weakening at the level of subcomponents, and it prevents the component re-use proposed above.

There are many directions for further research:

- Since the Hoare rules seem to be independent of the definitions of lr , lt , sr , and st , it may be that the language and logic could be generalised to have a more abstract construct that takes two subcomponents and combines the gain and phase values according to arbitrary functions f_r and f_θ (arbitrary to the extent of mapping two gain/phase values to a new gain and phase value, respectively). Or it might be desirable to restrict the functions so that they obey whatever algebra(s) lr , lt , sr , and st obey.
- It might also be possible to have Hoare rules defined over signals instead of over components. This might avoid the need to convert general block diagrams into a tree-structured language like Cosy, but it is not

immediately clear whether such a logic could represent the feedback in loops.

- If it is necessary to transform general block diagrams for control systems into a more constrained representation in order for Hoare logic to be used, then there remains the question of whether this can be done, and, if so, how. Along with this, is the question of how logical assertions can be included in block diagrams.
- The theory of Laplace transforms is well established, and we have simply assumed it here, as we have no reason to doubt the results. Nevertheless, for a complete formalisation in a mechanised theorem prover, it might be desirable to develop the theory from first principles, and prove the required results as theorems, instead of taking them as assumptions.
- Ultimately, the aim is to develop a similar approach for discrete-time control systems. Amongst other things, that would allow reasoning about the relationship between a continuous-time model and its discrete-time equivalent.

Acknowledgements

We are indebted to Manuela Bujorianu, John Hall, Rick Hyde, and Yoge Patel for sharing with us their insights into control engineering, and also to Rob Arthan, Tom Kelsey, and Colin O'Halloran for helpful discussions about the work.

References

- [1] M Arbib and E Manes Machines in a category SIAM review 57 (1974), 163-192
- [2] D Dill A theory of timed automata Theoretical Computer Science , 126(2):183-235,1994
- [3] R. Arthan, P. Caseley, C. O'Halloran, and A. Smith. ClawZ: Control laws in Z. In *Proc. 3rd IEEE International Conference on Formal Engineering Methods (ICFEM 2000)*, York, September 2000.
- [4] Martin Dunstan et al Lightweight formal methods for computer algebra systems ISSAC'98, ACM Press, 1998
- [5] Martin Dunstan et al Formal Methods for Extensions to CAS FM'99, LNCS 1709, Springer 1999

- [6] A Edalat and A Lieutier Domain theory and differential calculus Proc IEEE LICS 17, IEEE Press 2002
- [7] M. J. C. Gordon. Mechanizing programming logics in higher order logic. In G. Birtwistle and P. A. Subrahmanyam, editors, *Current Trends in Hardware Verification and Automated Theorem Proving*, pages 387–439. Springer-Verlag, 1989.
- [8] J Harrison Theorem proving in the real numbers Cambridge University Press, 1995
- [9] M. Hasegawa Models of Sharing Graphs Springer 1997
- [10] C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 583, October 1969.
- [11] Bruce Krogh Approximating Hybrid System Dynamics for Analysis and Control HSCC 1999, LNCS 1569, Springer, 1999
- [12] M Jirstrand Nonlinear Control System Design by Quantifier Elimination J Symbolic Comput., 24, 137-152, 1997.
- [13] The MathWorks. Simulink. <http://www.mathworks.com/products/simulink/>
- [14] B. Mahony. The DOVE approach to the design of complex dynamic processes. In *Proc. of the First International Workshop on Formalising Continuous Mathematics*, NASA conference publication NASA/CP-2002-211736, pages 167–187, August 2002.
- [15] Tobias Nipkow Hoare Logics in Isabelle/HOL. In *Proof and System-Reliability*, pages 341-367, Kluwer, 2002
- [16] K. Ogata. *Modern Control Engineering*. Prentice-Hall, third edition, 1997.
- [17] R. W. Pratt, editor. *Flight Control Systems: Practical Issues in Design and Implementation*, volume 57 of *IEE Control Engineering Series*. The Institution of Electrical Engineers, 2000.
- [18] Ashish Tiwari and Gaurav Khanna Series of abstractions for hybrid automata. In *Proc 5th International Workshop on Hybrid Systems: Computation and Control HSCC 2002*, LNCS 2289, Springer 2002
- [19] C Gurr and K Turlas Towards the principled design of software engineering diagrams. In *Proc. 22nd International Conference on Software Engineering* pages 509-520, ACM Press 2000.