The Syntax and Semantics of $\mu$-Charts

Greg Reeve and Steve Reeves

Department of Computer Science,
University of Waikato,
New Zealand
{gregr,stever}@cs.waikato.ac.nz

This document is a reduced version of Working paper 04/2004
Abstract. $\mu$-Charts is a language for specifying the behaviour of reactive systems. The language is a simplified variant of the well-known language Statecharts that was introduced by Harel [1]. Development of the $\mu$-Charts language is ongoing research undertaken under the auspices of the Formal Methods Laboratory of the Computer Science Department, University of Waikato [5]. This paper gives a comprehensive treatment of the syntax and semantic definition for $\mu$-Charts.

1 Introduction

The language StateCharts was introduced by Harel in [1]. Because Statecharts are based on finite-state automata, which are easily represented in a graphical form, they were the first of many “visual” specification languages. The ability to use diagrams to represent specifications has made Statecharts a popular specification tool for software engineers. The invention of Statecharts marked the beginning of extensive research into making precise their semantics, originally given informally in [1]. Some of this research led to the creation of another similar specification language for reactive systems called $\mu$-Charts.

The rather long pedigree of $\mu$-Charts is a chain of formalisms starting with Statecharts, as described above, and includes Mini-Statecharts [6], an extended version of Mini-Statecharts [13], the $\mu$-Charts of [7], the $\mu$-Charts of [12] and the $\mu$-Charts of [9].

More specifically $\mu$-Charts was originally presented by Philipp and Scholz in [7, 8] where they give their first semantics for the language. Later in [12], Scholz continued to develop $\mu$-Charts. Whilst making substantial changes to the first semantics, Scholz gives some refinement rules and a notion of implementation of charts. Subsequently a translation method from $\mu$-Charts into the $Z$ specification language was developed [11, 10, 9] in conjunction with the then ISuRF and later $Z\lambda$ research projects [5].

Previously, the most detailed description of the $Z$ translation of $\mu$-Charts was given in [10]. Here the authors showed both how to extend the original semantics for $\mu$-Charts to include local variables and value-carrying signals and how the $Z$ translation evolved to cope with this additional machinery.

This paper reflects work that has been ongoing since then. In some respects it is both a step forward and a step backwards from the last presentation of the charts to $Z$ translation [10]. Here we present, again, the latest syntactic and semantic definition for the $\mu$-Charts language. Unlike previously, we consider the presented translation of charts into $Z$ not as another encoding of “the $\mu$-Charts semantics” but as the semantics itself. Also, we give the semantic encoding or translation in its entirety rather than via examples. The semantics relates more closely (though not exactly) to the later definition by Scholz [12] rather than the original semantic definition of Philipp and Scholz. We reiterate, however, that our intention is not to compare this semantics with that, but simply to state the semantics that we consider as the current meaning for $\mu$-Charts.

The semantics we give here do not include local variables and value-carrying signals. It is however the basis for several current projects, one of which is the
proper integration of local variables and value-carrying signals. Others include the development of a refinement calculus and the ongoing development of tool support for the language.

In section 2 we present the definitions of the text-based syntax for charts in conjunction with demonstrating their related graphical presentation. Section 9 gives the semantics of the language. The proofs of the lemmas and propositions of Section 9 are given in a fuller version of this paper, available from the Waikato FM Lab web site www.cs.waikato.ac.nz/Research/fm/.
2 The Syntax of $\mu$-Charts

This section defines the syntax that we intend for $\mu$-Charts discussed in this paper. We present a textual syntactic definition in parallel with describing the graphical representation of the language. First we define the basic building block for $\mu$-Charts specifications which are sequential charts. Then we introduce the syntactic operators that can be used to combine sequential charts into more sophisticated $\mu$-charts. In particular we give the operators for allowing the composition of two charts, the decomposition of a chart, i.e. embedding an arbitrary chart into a state of a sequential chart, and the interface operator that allows a chart designer to specify the context of the chart, that is the signals with which a chart interacts with its environment. We make the point that each of these are important “first-class” syntactic operators in the language despite the ability to consider them semantically as syntactic sugar.

We formally define the textual syntax of the language using a phrase structure grammar. The graphical syntax is presented using examples.

The set of all possible syntactically correct $\mu$-charts is given by the set $\mu$-Charts as defined by the following grammar.\footnote{The phrase structure grammar presented here uses the bracket notation [...] to represent the optional inclusion of the term enclosed in the bracket. Also, the tokens of $\mu$-Charts are given in bold font.}

\[
\begin{align*}
\mu\text{Charts} & ::= \text{Charts} \mid ( \text{Charts} ) \\
\text{Charts} & ::= \mu\text{Chart} \mid \mu\text{CompChart} \\
& \quad \mid \mu\text{DecChart} \mid \mu\text{Chartio}
\end{align*}
\]

In the following we give productions for the non-terminals $\mu$Chart, $\mu$CompChart, $\mu$DecChart and $\mu$Chartio.

3 Sequential $\mu$-Charts

The smallest unit of $\mu$-Charts is a sequential $\mu$-chart. In general a simple sequential chart is the specification of the allowable behaviour of a named finite state automaton that takes input from its environment and instantaneously produces output accordingly.

A sequential chart contains the name of the automaton, a nonempty set of control states, including a special start state (denoted by a double outline in the graphical notation), a feedback set (described in Section 4), and labelled transitions between states. For example consider the chart pictured in figure 1(a).
AChart\_x\_y
\[\begin{array}{c}
\text{A} \\
in_x / out_y \\
\text{B}
\end{array}\]
Fig. 1.

This is an example of a sequential chart named \textit{AChart\_x\_y}. It contains two control states A and B, of which A is uniquely identified as the initial state. The chart contains one transition which has the guard \textit{in}_x and action \textit{out}_y. Informally we could describe this transition as follows: “if the chart \textit{AChart\_x\_y} is in state A and the current input event contains the signal \textit{in}_x then we instantaneously move to control state B outputing signal \textit{out}_y”.

Notice that we have also introduced a naming convention for charts. Due to the form of the name \textit{AChart\_x\_y} it is understood that the chart is “parameterized” on the labels \textit{x} and \textit{y}. Because we never use or refer to the abstraction \textit{AChart} itself, it is not necessary to define this mechanism in terms of usual abstraction, that is using variables, definitions and instantiations. Rather we assume the convention that, when given a chart definition such as \textit{AChart\_x\_y}, we are free to consider the chart \textit{AChart\_a\_b} with the understanding that it is defined as \textit{AChart\_x\_y} where all occurrences of \textit{x} are replaced by \textit{a} and similarly \textit{y} by \textit{b}. The chart \textit{AChart\_a\_b} is pictured in Figure 1(b).

To describe the attributes of charts formally, we introduce the following abstract sets.

\[[\text{Signal}, \text{Name}, \text{State}, \text{Param}]\]

Each of these sets contain labels used to identify signals, chart names, control states, variables and parameters respectively.

We also introduce the sets \textit{Chartname} and \textit{\(\mu\)Signal} whose elements are defined using the following grammar.

\[
\text{Chartname} ::= \text{class-name} \_ \text{param-list} | \text{class-name}
\]

\[
\text{\(\mu\)Signal} ::= \text{signal-name} \_ \text{param-list} | \text{signal-name}
\]

\[
\text{param-list} ::= \text{Param} [ \_ \text{param-list}]
\]

\[
\text{class-name} ::= \text{Name}
\]

\[
\text{signal-name} ::= \text{Signal}
\]

Now, consider all possible reactive systems that we might ever want or be able to specify using a \(\mu\)-chart. The set \textit{Chartname} is the set of labels used to name sequential charts. The set \(\mu\text{Signal}\) contains labels that name (possible parameterised) signals that these reactive systems can 1) use to interact with their environment, and 2) use for internal communication between their components.\(^2\) Similarly, \textit{State} is the set of labels used to name control states.

\(^2\) Again, context permitting, we use the elements of each of these sets to refer to both the label itself and the object that the label represents.
Each sequential chart contains a set of transitions. Each transition is a member of the set $\mu Transition$ and is made up of a tuple of the form $(S_f, S_t, label)$ such that $S_f$ represents the state from which the transition originated, $S_t$ represents the destination state of the transition and $label$ defines under what conditions the transition can be taken and the resulting action of that transition. The label of the transition is an element of the set $Transition$. We define the necessary sets as follows.

$$
\mu Transition \ ::= \ ( State , State , Transition )
$$

$$
Transition \ ::= \ guard / action
$$

$$
guard \ ::= \ signal-expr \ [ \&\ guard ]
$$

$$
signal-expr \ ::= \ [ - ] Signal
$$

$$
action \ ::= \ \{ signal-list \}
$$

$$
signal-list \ ::= \ Signal \ [ , signal-list ]
$$

Now we can define the set $\mu Chart$ of all allowable sequential charts.

$$
\mu Chart \ ::= \ ( Chartname , control-states , initial-state ,
\hspace{1cm} feedback-set , transitions [ , signal-list ] )
$$

$$
control-states \ ::= \ \{ \ initial-state [ , states-list ] \}
$$

$$
initial-state \ ::= \ State
$$

$$
states-list \ ::= \ State \ [ , states-list ]
$$

$$
feedback-set \ ::= \ \{ [signal-list] \}
$$

$$
transitions \ ::= \ \{ [transition-list] \}
$$

$$
transition-list \ ::= \ \mu Transition \ [ , transition-list ]
$$

According to this definition sequential charts are described as a tuple that has the fields $(C, \Sigma, \sigma, \Psi, \delta)$ such that

- $C$ is the name of the chart
- $\Sigma$ represents the set of control states
- $\sigma$ names the initial state
- $\Psi$ represents the set of feedback signals
- $\delta$ represents the set of transitions that are defined in the chart

Notice that every sequential chart must have at least one state (the initial state) but can have an empty set of feedback signals and transitions.

Each sequential chart has an input and output interface. These interfaces represent the sets of signals that the chart can react to and output respectively. From the definition of a chart (either textual or graphical) we can typically derive these interfaces from the signals that appear in the triggers of transitions and the signals that appear in the actions of the transitions respectively. Henceforth, we refer to these derived interfaces as the natural input / output interface of a chart. We do so because it proves useful if the designer of the chart can specify the interfaces of a chart that differ from the natural interface.
We introduce three auxiliary syntactic functions that are often used in conjunction with the syntactic definition of charts.

\[ \text{in}_s : \mu \text{Chart} \rightarrow \mathbb{P} \mu \text{Signal} \]
\[ \text{out}_s : \mu \text{Chart} \rightarrow \mathbb{P} \mu \text{Signal} \]
\[ \omega : \text{Chartname} \rightarrow \mu \text{Charts} \]

As mentioned, in the absence of a specified input / output interface for a sequential chart \( C \), we can calculate these interfaces using the functions \( \text{in}_s \) and \( \text{out}_s \) respectively. These functions simply return the natural (or obvious) interfaces for the chart. We give general versions of these functions in Section 8. We also adopt another notational convenience by assuming \( \text{in}_C = \text{def} \text{in}_s C \) and similarly \( \text{out}_C = \text{def} \text{out}_s C \).

The third function \( \omega \) when applied to a name, say \( C \), returns the chart with the name \( C \). We assume this function is always defined to have the property that for any sequential chart \( \omega C = (C, \Sigma, \sigma, \Psi, \delta) \) for appropriate \( \Sigma, \sigma, \Psi \) and \( \delta \). In fact, while this function is technically necessary, in practice the name of a chart, say \( C \), is often used where the intention is \( \omega C \). We use this shorthand freely assuming the context makes it clear which we are referring to.

There is a clear relation between the textual form of the syntax that is presented here and the graphical presentation of sequential charts, for example Figure 1(b). The chart name is presented in the upper-left corner of the chart and each of the control state are labelled ovals etc.. However it is worth noting that the textual definition of charts does not suppose to encode any spatial layout of a graphical chart. Hence, for any “textual chart” there are a great number of equivalent but different looking “graphical charts”.

The description of simple sequential charts so far has assumed that the input and output of a chart do not interact. It is, however, possible that a sequential chart contain a feedback loop that causes some of the output of the chart to be instantaneously fed back and act as input. As introduced, the description of sequential charts contains a feedback set \( \Psi \). This is used to specify which output signals can be instantaneously fed back and hence act as input. The following section introduces sequential charts with feedback.

4 \( \mu \)-Charts that feed back Signals

The feedback set in sequential charts allows us to model the instantaneous feedback of designated output signals in sequential charts. If a signal is in the feedback set then whenever that signal is output by the chart it is also instantaneously available as input. If we consider the input / output behaviour that a sequential chart describes as a back box that has a wire representing input and a wire representing output then the feedback of signals can be considered as a loop-back wire that connects the output to the input and carries only the designated signals.

In the graphical representation of charts the feedback set is presented in a box attached to the outline that encloses the chart. Because sequential charts
often have no associated feedback, i.e. their feedback set is empty, the graphical representation, as demonstrated in Figure 1(a), allows such simple sequential charts to be drawn without any feedback box attached. On the other hand, Figure 2 is an example of a sequential chart $AChart_{a\_b}$ that feeds back all of its signals, i.e. it has the feedback set containing $in_a$ and $out_b$.

![Fig. 2.](image)

### 5 Composition of $\mu$-charts

Now that we have introduced the basic building block of the language, i.e. sequential charts, we present some operators that can be used to combine arbitrary charts into more sophisticated $\mu$-charts.

Given any two charts $C_1 \in \muCharts$ and $C_2 \in \muCharts$ and set of signals $\Psi$, we denote the composition of these charts in parallel by an expression of the form $C_1 \mid \Psi \mid C_2$. The set of all such charts formed using the composition operator is defined by the set $\muCompChart$.

$$\muCompChart ::= \muCharts \{ \text{signal-list} \} \muCharts$$

From this definition we can see, as one might expect, the feedback of signals is intimately coupled with the composition of charts. Hence when we compose two charts the signals that they are able to communicate on are included as part of the composition operator. The feedback set can be empty, in which case the composition would represent two charts that proceed in lock-step but do not affect one another in any way.

The graphical notation again uses two conventions for the treatment of feedback between composed charts. Consider the two composed charts in Figure 3.

![Fig. 3.](image)
The chart in Figure 3(a) represents the chart $A\text{Chart}_{a.\ b} | \{} | B\text{Chart}_{b}$ where $A\text{Chart}_{a.\ b}$ is $(A\text{Chart}_{a.\ b}, \{A, B\}, \{\}, \{(A, B, in_a/out_b)\})$ and $B\text{Chart}_{b}$ is similarly defined. The chart of Figure 3(b) represents the chart $A\text{Chart}_{a.\ b} | \{} | B\text{Chart}_{b}$.

6 The Decomposition of a $\mu$-chart

A decomposed $\mu$-chart refers to a sequential chart that has other charts embedded into some or possibly all of its states. The purpose of decomposing a state in a sequential chart is to mimic a master / slave relationship between the charts. The sequential chart that has its states decomposed can be considered the master in the relationship because it effectively allows the decomposing chart, i.e. the slave, the ability to react to input from the environment only when the master is in the state that the slave decomposes. Of course a master can have several slaves. A slave, however, has only one master which is always a sequential chart. The master may itself be a slave, i.e. there can be more than one level of decomposition.

The syntax for decomposition is defined by the following grammar.

$$
\mu\text{DecChart ::= Dec } \mu\text{Chart by } \{ \text{ dec-states } \}
$$

$$
\text{dec-states ::= ( State, } \mu\text{Charts ) [ , dec-states ]}
$$

In the graphical representation of a decomposed chart the decomposed states of the master are written as rectangles (rather than the usual oval state representation). The slave chart is then given in a separate chart diagram. Figure 4 presents an example of a decomposed chart given by the textual expression

$$
Dec (A\text{Chart}_{a.\ b}, \{A, B\text{Chart}_{b}\}, A, \{(A, B\text{Chart}_{b}, in_a/out_b)\})
$$

by $(B\text{Chart}_{b}, (B\text{Chart}_{b}, \{C, D\}, C, \{(C, D, out_b/sigc)\}))$

![Fig. 4.](image)

7 The Interface Operator

We refer to the final chart operator that we introduce as the interface operator. As we discussed above each sequential chart has a natural input / output interface associated with it. That is, the signals that occur on the triggers of transitions together make up the input interface while the signals that appear in the actions of transitions form the natural output interface.
However, it is often the case that the natural interfaces of a chart are not those that are intended by the designer. The most common situations where explicit interfaces are required is when a chart designer wishes to hide signals. A complex specification is (in general) constructed by combining several sub-charts using the operators that we have already introduced in this section. We can consider each sub-chart as sharing an observable interface with its environment. For example consider the composition (with feedback) of the sequential charts $A_{chart_a \cdot b}$ and $B_{chart_b}$ pictured in Figure 3(b). The “effective” environment, i.e. what a chart sees and allows to be seen, for the respective charts $A_{chart_a \cdot b}$, $B_{chart_b}$ and $A_{chart_a \cdot b} \mid \{out_b\} \mid B_{chart_b}$ are all different things. It is usually the case that we wish to distinguish between the way that one part of the specification, e.g. $A_{chart_a \cdot b}$, interacts with its environment and the way that the overall specification, e.g. $A_{chart_a \cdot b} \mid \{out_b\} \mid B_{chart_b}$ interacts with its environment. For instance it may be the case that in this example we want the signal $out_b$ to be used only for internal communication between $A_{chart_a \cdot b}$ and $B_{chart_b}$. Hence the interface that these charts share individually must contain signal $out_b$, however we want to hide this signal from the observable interface of the specification $A_{chart_a \cdot b} \mid \{out_b\} \mid B_{chart_b}$. This can be achieved using the interface operator.

Another way that the interface operator can be used is to specify that the output interface of the chart is larger than its natural output interface. The reason a designer would wish to do this becomes clearer when we consider the entire definition of the language. In particular, the refinement calculus for $\mu$-Charts. In brief, the $\mu$-Charts language makes the assumption that the output interface of a chart (whether left unspecified, i.e. the natural interface, or explicitly defined) specifies the context in which the chart designer expects the reactive system to reside. The refinement calculus for charts allows us to extend this output context using refinement. Such refinements allow new output behaviour over the addition signals of the new context. Hence, specifying an output context that is larger than the natural output interface of the chart allows a designer to indicate that a chart chooses not to output some signals that control its environment.

As we have seen, there are two interfaces for any chart, the input interface and the output interface. The interface operator allows us to explicitly specify either or both of these sets of signals. Restricting the input interface, i.e. defining an explicit input interface that is a subset of the natural interface can be considered as filtering input from the environment, that is, these filtered signals are never received as input from the environment. Restricting the output interface means the hidden signals, i.e. signals that are in the natural output interface but not in the explicit output interface, are not observable as output from the specified reactive system.

The set of syntactically correct charts that contain hiding are given by $\mu_{chartio}$ which is defined as follows.
\[ \mu_{\text{Chart}io} ::= [\text{input} I] \mu_{\text{Charts}} [\text{output} I] \]

\[ \text{inputI} ::= \{ \text{signal-list} \} \]

\[ \text{outputI} ::= \{ \text{signal-list} \} \]

\[ \text{signal-list} ::= \text{Signal} [, \text{signal-list}] \]

Notice that both the explicit input interface (on the left) and the explicit output interface (right) are optional. This allows the obvious shorthand notation such that \( x[C] =_{\text{def}} x[C]_{\text{out}C} \), and \( C[Y] =_{\text{def}} \psi[C]_{\text{in}C} \).

The graphical notation includes the explicit definition of interfaces in the box attached for feedback. The feedback box for a chart that employs the interface operator can be parsed as “in = input-interface I feedback-set I out = output-interface”.

For example, the chart \( \{in.a|A\text{Chart}_{a\_b} | \{out\_b\} | B\text{Chart}_{b}\} \) pictured in Figure 5 hides the signal \( out\_b \) from the input interface of the chart. This means that the only way that chart \( B\text{Chart}_{b} \) can be affected by the input signal \( out\_b \) is if that signal is output from the chart \( A\text{Chart}_{a\_b} \). Note however, that the signal \( out\_b \) is still observable output from this specification.

![Diagram](image)

Fig. 5.

Again, we define some common shorthand notation that is useful when using the graphical notation for charts. Given \( inC \) and \( outC \) are the respective natural input / output interfaces for some chart \( C \) then we are free to use the following shorthand conventions.

\[
\begin{align*}
\text{in} = t \ 1 \ \Psi & =_{\text{def}} \text{in} = t \ 1 \ \Psi \ | \text{out} = outC \\
\text{out} = \ 0 \ \Psi & =_{\text{def}} \text{in} = inC \ 1 \ \Psi \ | \text{out} = \ 0 \\
\Psi & =_{\text{def}} \text{in} = inC \ 1 \ \Psi \ | \text{out} = outC \\
\text{in} = t \ \text{out} = \ 0 & =_{\text{def}} \text{in} = t \ 0 \ | \text{out} = \ 0
\end{align*}
\]

where \( \Psi, t, \) and \( o \) represent arbitrary sets of signals.

8 Auxiliary semantic functions

As for sequential charts we often define the useful auxiliary function

\[ \omega : \text{Chartname} \rightarrow \mu_{\text{Charts}} \]
for more complex charts. We often introduce a complex $\mu$-Charts specification and give it a simpler name, for example the expression $C_{12} = C_1 \mid F \mid C_2$ defines the function $\omega$ such that $\omega C_{12} =_{def} \omega C_1 \mid F \mid \omega C_2$. Again we generally use the name $C_{12}$ to refer to the chart $\omega C_{12}$.

Finally, we give the general definition for two of the syntactic functions, $\text{in}$ and $\text{out}$, that were introduced for sequential charts (as $\text{in}_s$ and $\text{out}_s$) in Section 3.

\[
\begin{align*}
\text{in} : \mu\text{Charts} &\rightarrow \mathbb{P}\mu\text{Signal} \\
\text{in} (C, \Sigma, \sigma, \Psi, \delta) &= \text{in}_s (C, \Sigma, \sigma, \Psi, \delta) \\
\text{in} (C_1 \mid \Psi \mid C_2) &= \text{in} C_1 \cup \text{in} C_2 \\
\text{in} (\text{Dec}_C \text{ by Slaves}) &= \\
&\quad \text{in} C \cup \left(\bigcup\{(\lambda(sn, \text{Slave}) \bullet \text{in} \text{Slave}) \mid (sn, \text{Slave}) \in \text{Slaves}\}\right) \\
\text{in} (i[C]) &= I
\end{align*}
\]

\[
\begin{align*}
\text{out} : \mu\text{Charts} &\rightarrow \mathbb{P}\mu\text{Signal} \\
\text{out} (C, \Sigma, \sigma, \delta) &= \text{out}_s (C, \Sigma, \sigma, \delta) \\
\text{out} (C_1 \mid \Psi \mid C_2) &= \text{out} C_1 \cup \text{out} C_2 \\
\text{out} (\text{Dec}_C \text{ by Slaves}) &= \\
&\quad \text{out} C \cup \left(\bigcup\{(\lambda(sn, \text{Slave}) \bullet \text{out} \text{Slave}) \mid (sn, \text{Slave}) \in \text{Slaves}\}\right) \\
\text{out} (\underline{C}[O]) &= O
\end{align*}
\]
9 Language semantics

Our formal semantics for $\mu$-Charts is given by a method for constructing a Z model for an arbitrary chart. The kernel logic $Z_C$ for Z, presented in [4], gives the Z model a meaning grounded in typed set theory. Hence, the combination of these gives us a semantics for $\mu$-Charts grounded in typed set theory.

As we have already seen a $\mu$-chart, specifying the behaviour of a reactive system, can be (and in most interesting cases is) made up by using language operators to combine simple sequential charts into more complex specifications. We present the semantic by giving an informal account of the meaning of the language constructs in parallel with describing the general method for creating the Z model and hence the formal semantics.

A simplifying assumption made in the semantics for $\mu$-charts is that all constituent sequential sub-charts, that are combined to describe a reactive system, proceed in lock-step. When a step happens is determined by the environment in which the chart resides. That is, when the environment produces input each sequential chart makes a transition. Notice that we distinguish between an input signal and input (or an input event), which is a set of input signals.

Initially, we give a step semantics for charts. This describes the behaviour of a chart in terms of the output that the chart produces in response to input from the environment assuming a given state. The step semantics essentially relates the current configuration of a chart and input to a new configuration and the resulting output. This relation describes every possible step that a chart can take.

We will show later how this step semantics can be lifted to give a trace semantics that abstracts away the information about the charts configuration. This gives the meaning of charts in terms of sets of observable input / output traces that an implementation of the specification can exhibit over time. The trace semantics is given primarily to relate the account of $\mu$-Charts refinement given here back to the previous work describing refinement. The notion of chart refinement presented here is characterized in terms of the step semantics. Hence it is not strictly necessary to have the trace semantics at all, though the trace semantics helps to give a more intuitive understanding of the resulting refinement relation.

Building the Z model for charts requires two separate tasks. The first part of the process is to create a general model for each of the separate constituent sub-charts in the $\mu$-chart being described. We call this the transition model. As one might expect given the language, this process is recursive in nature and follows exactly the structure of the chart being modeled. The overall transition model is built up by initially describing the model for each of the sequential sub-charts followed by the description of each of the new sub-charts created using the language operators. The final model describes the largest sub-chart which is

---

$^3$ The term configuration is used to refer to the “state” of a chart because a $\mu$-chart that has several sequential sub-charts can be in several states at the same time, in other words each sequential sub-chart has its own current state.
the specification of the reactive system of interest. The modular nature of the language is such that the model of an arbitrary sub-chart can be consider as a “black-box”. The following diagram demonstrates both the informal “circuit diagrams” that we use to motivate the semantic description and the “black-box” characteristics of the transition model for an arbitrary chart $C$.

![Diagram of a chart $C$ with inputs $i$, output $o$, and active state]

The circuit diagram gives the structure of the general transition model for a chart. That is, we can consider the model of any chart with arbitrary structure as a circuit that has two inputs and one output. The chart model defines a function of output given the two inputs.

Sections 10 to 15 give both an informal description and a formal treatment that introduces the transition model for each of the language constructs. The formal treatment has two parts. The first gives a generic process for creating a Z model for an arbitrary $\mu$-chart. The Z description of the chart can be used in the standard Z fashion to investigate the behaviour of the chart. Using available proof assistant and Z animation tools is a useful way to perform some validation of the $\mu$-chart’s behaviour. The second part of the formal treatment uses the meaning of the Z itself, via the Z logic of [4], to give a logic for reasoning about $\mu$-charts. This logic is presented in terms of introduction and elimination rules for each of the language constructs. It can be used to reason about the model of a $\mu$-chart. The rules are presented in a natural deduction style, closely following the work of Deutsch, Henson and Reeves [2, 3].

The final step (Section 16) in building the model for a $\mu$-chart is to hide the machinery that is present in the transition model, that is the machinery that is necessary to model the inherent structure of charts. We refer to this model of a chart as the step semantics or partial relations semantics for a chart.

10 Sequential charts

A sequential chart is essentially a finite state automaton that describes the set of output signals that results from reacting to a set of inputs in a given state. The general transition model of a sequential chart has two notions of state. The first being the usual automaton notion of state which is determined by the transitions that have happened due to input since the automaton was initialized. This state is denoted in a chart diagram by labeled ovals or rectangles. The second notion of state represents whether or not the sequential chart is active. The reason that a chart can be active or inactive is because it may be the slave of another chart in the specification via decomposition. Finally, a sequential chart
can instantaneously feed back output signals. This means that, during any step of the chart, feedback output is also considered as input.

The structure of the transition model of a sequential chart \((C, \Sigma, \sigma; \Psi, \delta)\) is demonstrated by the following circuit diagram.

\[
\begin{array}{c}
\text{active} \\
\Sigma, \delta \\
\Psi \\
o
\end{array}
\]

This diagram can be separated into three logical parts as described above. The finite state automaton specification of the input/output relationship is denoted by the component labeled \(\Sigma, \delta\). The active/inactive state mechanism is denoted by the input labeled \(\text{active}\). And the feedback mechanism is represented by the line labeled \(\Psi\).

Consider the chart \((C, \{A, B\}, A, \{\}, \delta)\) (where \(\delta\) is the appropriate transition description) pictured in Figure 6.

![Fig. 6. A simple sequential \(\mu\)-chart](image)

We can clearly see the finite state description of the input and output behaviour from this typical sequential \(\mu\)-chart. Informally the behaviour that this chart captures can be described as: the chart starts in state \(A\); if it is in state \(A\) and the signal \(a\) is input then signal \(b\) is output and the chart changes to state \(B\); and similarly if it is in state \(B\) and \(c\) is in the input then \(d\) is output and the new state is \(A\). Notice that we have not yet described what happens when the actions on the transitions are not satisfied. Hence the term \textit{partial relations semantics}. We consider the meaning of a chart outside of its defined transitions after we introduce the Z semantics.

The ability of a sub-chart to be active or inactive is necessary for charts that are embedded (i.e., slaves of) some other chart in a decomposition. When a chart is active it reacts to input as specified by its finite state description. When it is inactive it ignores input, remains in the same state and gives no output. In fact, no output actually means the output is the empty set of signals.

The behaviour of the feedback mechanism is uninteresting in this example because there are no signals fed back. However, in general, a sequential chart can feed back its output signals which then instantaneously act as input. This is represented in the circuit diagram by the link labeled \(\Psi\) that creates a loop.
between output and input and is assumed to carry only the specified feedback signals. We discuss in more detail exactly what feedback in sequential charts means semantically in Section 12.

11 The Z transition model for sequential $\mu$-Charts

The essence of the general transition model for sequential $\mu$-charts is the description of each of the transitions in a sequential chart as a separate Z operation schema. Each of these operation schemas (one for each transition in the chart) are combined using Z schema disjunction to give one schema that describes the abstract transition behaviour of the sequential chart. The Z state of the model has an observation that determines the current state of the chart. The operation schema describing a transition describes how and when that state changes. We proceed by introducing the Z that is given as the transition semantics of a chart and then discussing the meaning of that Z.

For a general sequential chart $(C, \Sigma, \sigma_0, \Psi, \delta)$ we introduce the following definitions (left) that encode the chart’s states, input interface and output interface in Z.

$$\begin{align*}
\text{states}_C & : \mathbb{P} \mu\text{State} \\
\text{inc}_C & : \mathbb{P} \mu\text{Signal} \\
\text{out}_C & : \mathbb{P} \mu\text{Signal} \\
\Psi & : \mathbb{P} \mu\text{Signal} \\
\text{states}_C & = \Sigma \\
\text{inc}_C & = \text{in } C \\
\text{out}_C & = \text{out } C
\end{align*}$$

$$\begin{align*}
\text{Chart}_C & == [c_C : \text{states}_C] \\
\text{Init}_C & \\
\text{Chart}_C \\
\text{c}_C & = \sigma_0
\end{align*}$$

The general state schema $\text{Chart}_C$ (top right) models the automaton state for the chart $C$. And the initial state of the chart is modeled by the schema $\text{Init}_C$ (bottom right).

A separate state schema is also given for each automaton state in the chart. So for all $\sigma \in \Sigma$ there exists a schema such that

$$C \sigma == [\text{Chart}_C \mid c_C = \sigma]$$

Now we give an operation schema for each transition in the chart. That is for all $(S_f, S_t, \text{guard/action}) \in \delta$ we define an operation schema that has the following structure.
This general operation schema introduces a new syntactic function $\rho$ that produces the $Z$ predicate that models the syntactic representation of a transition’s guard. If we consider a transition’s guard in general as a (possibly empty) list of signal expressions separated by the symbol $\&$ then each of the elements in the list can be classified into two categories: either a positive signal expression—simply the name of a signal; or a negative signal expression—the signal name is prepended with a minus sign. A positive signal expression, say $\text{sig}$ where $\text{sig} \in \mu_{\text{Signal}}$, is denoted by the $Z$ expression $\text{sig} \in i_{C}? \cup (o_{C}! \cap \Psi)$. A negative signal expression, say $-\text{sig}$, is denoted by the $Z$ expression $\text{sig} \notin i_{C}? \cup (o_{C}! \cap \Psi)$. The function $\rho$ performs this transformation on each signal expression in the list and connects the resulting predicates back together using conjunction. If the list is empty the predicate produced by $\rho$ would be $\text{true}$.

This general scheme for giving the $Z$ for a transition defines the semantic function $[\llbracket \cdot \rrbracket_{Z_{t}}$ such that for an arbitrary transition $(S_{f}, S_{t}, \text{guard/action})$

$$[\llbracket (S_{f}, S_{t}, \text{guard/action}) \rrbracket_{Z_{t}} =_{\text{def}} \delta_{S_{f}, S_{t}}$$

where the schema $\delta_{S_{f}, S_{t}}$ results from the method described above.

Along with the schemas for each transition we also need a single schema that models the behaviour of the chart when it is inactive. Again for the general sequential chart $(C, \Sigma, \sigma_{0}, \Psi, \delta)$, the inactive schema is given as follows.

$$\text{Inactive}_{C}$$

$$\begin{align*}
\exists \text{Chart}_{C} \\
i_{C}? : P \text{ in}_{C} \\
\text{active}_{\mu} : P \mu_{\text{State}} \\
o_{C}! : P \text{ out}_{C} \\
\neg \text{active}(C) \\
o_{C}! = \{\} 
\end{align*}$$

The transition semantic of a sequential chart can now be given by the following definition.$^{4}$

$$[\llbracket (C, \Sigma, \sigma_{0}, \Psi, \delta) \rrbracket_{Z} =_{\text{def}} \delta_{C}$$

$^{4}$ We use the notation $\bigvee X$ to denote the schema disjunction of all the schemas in the set $X$. 

17
where
\[ \delta_C = (\forall \{t\}_{Z_i} \mid t \in \delta) \lor \text{Inactive}_C \]

This concludes the presentation of the Z that is used to give the transition model for a sequential chart. We now consider the meaning of this Z in terms of the particular example that is pictured in Figure 6.

The following Z schemas result from giving the transition model of this chart.\(^5\)

**\(\delta_{AB}\)**
- CA
- CB'
- \(i_C?: \mathbb{P} \text{in}_C\)
- \(\text{active}_- : \mathbb{P} \mu_{\text{State}}\)
- \(o_C!: \mathbb{P} \text{out}_C\)
- \(\text{active}(C)\)
- \(a \in i_C? \cup (o_C! \cap \Psi)\)
- \(o_C! = \{b\}\)

**\(\text{Inactive}_C\)**
- \(\mathbb{Z} \text{Chart}_C\)
- \(i_C?: \mathbb{P} \text{in}_C\)
- \(\text{active}_- : \mathbb{P} \mu_{\text{State}}\)
- \(o_C!: \mathbb{P} \text{out}_C\)
- \(\neg \text{active}(C)\)
- \(o_C! = \{}\)

The transition in chart \(C\) from state \(A\) to \(B\) results in the Z schema \(\delta_{AB}\). Rather than describing the schema in terms of its syntax we proceed directly to describing its meaning that is given by the set \([\delta_{AB}]\). In the theory \(\mathbb{Z}_C\) [4] the meaning of a schema is given as a set of bindings. A binding is a mapping between the labeled observations \(l_i\) and their respective values \(t_i\) and is written as \(\langle | \ldots l_i \rightarrow t_i \ldots | \rangle\).

Hence the meaning of the schema \(\delta_{AB}\) is given by the following set comprehension.

\[
[\delta_{AB}] = \{ \langle | c_C \Rightarrow A, \ i_C? \Rightarrow i, \ \text{active}_- \Rightarrow \text{active}, \ c_C' \Rightarrow B, \ o_C! \Rightarrow \{b\} | \ |
\]

\[ i \subseteq \text{in}_C \land \text{active} \subseteq \mu_{\text{State}} \bullet C \in \text{active} \land a \in i \}

We can see from this definition that each binding in the semantic set has five labeled observations. The meaning of these are:

- \(c_C\)—the state of the chart before the transition happens, in this case the state \(A\)
- \(i_C?\)—the set of input signals which are offered by the environment that are in the input interface of the chart
- \(\text{active}_-\)—a set that denotes all currently active charts
- \(c_C'\)—the state of the chart after the transition happens, in this case state \(B\)
- \(o_C!\)—the output generated by this sequential chart, in this case the set containing the signal \(b\)

\(^5\)We assume that all of the entities given in the presented Z whose definitions are omitted have the appropriate type and obvious meaning.
Hence each transition description is “parameterized” on the current state of the chart \(c_C\), the input from the environment \(i_C?\) and whether or not this chart is active. If the precondition of the schema holds, i.e. the chart is active and the transition is triggered, then the output contributed by this sequential chart is defined by the set \(o_C!\). Note that in the schema itself the expression \((o_C! \cap \Psi)\) models the feedback mechanism for the sequential chart. In this case \(\Psi = \{\}\), that is the feedback set is empty. A similar schema to \(\delta_{AB}\) that models the transition from state \(B\) to \(A\) called \(\delta_{BA}\) would also be given.

The second schema \(\text{Inactive}_C\) describes the behaviour of the sequential chart when it is inactive. We can see from the set of bindings

\[
\{\{c_C \mapsto s, \ i_C? \mapsto i, \ \text{active} \mapsto \text{active}, \ c_C' \mapsto s, \ o_C! \mapsto \{\}\ \} \mid s \in \{A, B\} \land \text{active} \subseteq \mu_{\text{State}} \land i \subseteq \text{in}_C \land c_C' \notin \text{active}\}
\]

that the schema \(\text{Inactive}_C\) faithfully models the inactive behaviour of the chart as we described above.

Now the complete transition model for chart \(C\) is defined as the disjunction of each of the individual transition schemas to be \(\delta_C = \delta_{AB} \lor \delta_{BA} \lor \text{Inactive}_C\). Because of the definition of schema disjunction the set \(\llbracket \delta_C \rrbracket\) contains all of the bindings from the sets \(\llbracket \delta_{AB} \rrbracket\), \(\llbracket \delta_{BA} \rrbracket\) and \(\llbracket \text{Inactive}_C \rrbracket\).

This method of investigating the semantics of the transition model for charts, that is considering the meaning of schemas as sets of bindings, is somewhat informative in the simple example presented above. However, in general we do not want to be forced to resort to complex set definitions and manipulations to reason about the meaning of charts. Therefore, as discussed previously, we give a set of rules that characterise and allow us to reason about the Z semantic model for charts. These and other rules, presented later, give us a logic for charts.

Firstly, we formalize what it means for a binding of the transition model to satisfy a specific (syntactic) transition of the chart. Given an arbitrary transition of the form \(t = (S_f, S_t, \text{guard}/\text{action})\), from the chart \(C\), where \(\llbracket t \rrbracket^T\) we have,

\[
\text{Trans } t \ z = \omega z.\ C = t.S_f \land \rho(t.\text{guard})[\alpha T/z.\alpha T] \land z.\ c'_C = t.S_t \land z.\ o_C! = t.\text{action}
\]

The terms \(t.S_f\) etc. are assumed to be defined in the obvious way such that \(t.S_f\) gives the “from state” of a transition, \(t.\text{guard}\) gives the guard component of a transition, \(t.S_t\) gives the “to state” and \(t.\text{action}\) returns the action component. The function \(\rho\) is as defined above.

Now we give the formal definition of the transition model for charts directly in terms of the meaning of the Z model. For the arbitrary sequential chart \((C, \Sigma, \sigma_0, \Psi, \delta)\), we have,\(^6\)

\(^6\) We make the assumption that whenever a meta-variable is used to define the type of the transition model for a chart, e.g. \(T\), it represents the type of the schema as given in the Z model rather than the type of the schema after normalisation. For example the type \(T = [c_C, c'_C : \text{states}_C; i_C? : \text{P in}_C; \ \text{active}_C : \text{P} \mu_{\text{State}}; \ o_C! : \text{P out}_C]\)
\[[\delta C]^T\ =_{\text{def}} \{ z \in T \mid C \not\in z.\text{active} \land z \in T[\Sigma : \Sigma] \land z.\text{oC}! = \{\} \lor C \in z.\text{active} \land \exists t \in \delta \cdot \text{Trans} t z\} \]

From this definition we derive the following introduction and elimination rules\(^7\).

**Proposition 111**

\[
\begin{align*}
\text{active } C z &\quad P \quad \text{Trans } t z \vdash P (z_{i\rightarrow}) \\
\text{inactive } C z &\quad z \in \delta C \quad z.\text{oC}! = \{\} \quad (z_{i\rightarrow})
\end{align*}
\]

\[
\begin{align*}
\text{active } C z \quad \text{Trans } t z &\quad z \in \delta C \quad (z_{i\rightarrow}) \\
\text{inactive } C z &\quad z \in \delta C \quad z.\text{oC}! = \{\} \quad (z_{i\rightarrow})
\end{align*}
\]

where we assume the usual conditions for \( t \) and \( P \).

Proofs for each of these rules are trivial given the definition \([\delta C]\]. The predicates \( \text{active } C z \) and \( \text{inactive } C z \) are defined as,

\[
\begin{align*}
\text{active } C z &\ =_{\text{def}} C \in z.\text{active} \\
\text{inactive } C z &\ =_{\text{def}} \neg \text{active } C z
\end{align*}
\]

for sequential charts.

Given these rules we can start to investigate some of the properties of the transition model. In particular, the following propositions allow us to reason about which bindings are in the transition model and which are not.

Firstly, we make the observation that the transition model given makes a distinction between two sources of input that can contribute to the signals that trigger a transition. The external input to the chart generated by the environment \( i_C ? \) and the fed back output from the chart itself, \( i.e. \) for the chart \((C, \Sigma, \sigma_0, \Psi, \delta)\) the set denoted by \( o_C! \cap \Psi \). We formalize this observation in the following proposition.

**Proposition 112** For the arbitrary sequential chart \((C, \Sigma, \sigma_0, \Psi, \delta)\), \( z \in T \) and \( x \in T \),

\[
\begin{align*}
z \in \delta C &\quad z.\text{active } C z \cup \text{fb } z = x.\text{active } C z \cup \text{fb } x &\quad (z_{i\rightarrow}) \\
\end{align*}
\]

where \([\delta C]^T\) and \( T_i = T - [i_C ? : \text{P } \mu \text{Signal etc.}] \)

rather than the alternative in which the observation \( i_C ? \) would have the normalised type \( \text{P } \mu \text{Signal etc.} \).

\(^7\) Context permitting we omit the semantic brackets \( [\ ] \), so \( z \in \delta C \) is shorthand for \( z \in [\delta C] \).
The function $fb$ is a shorthand for representing the feedback set that is applicable to a chart. For the sequential chart $C$ and the arbitrary binding $z \in T$, as above,

$$fb \ z = \text{def} \ z \cdot \sigma_C! \cap \Psi$$

Here we proved the rule $(Z^\xi)$ just for sequential charts. In the following sections we prove the same rule for charts that are built using each of the chart operators. Hence these rules hold for charts in general regardless of their structure.

12 Feedback of signals in $\mu$-Charts

The semantics of feedback in sequential charts is encoded in the transition model as given in the previous section. In this section we give two pathological examples that demonstrate exactly the semantics of sequential charts with feedback.

The examples $C_1 = (C_1, \Sigma, \sigma, \{a\}, \delta_1)$ and $C_2 = (C_2, \Sigma, \sigma, \{a\}, \delta_2)$ (for appropriate $\Sigma, \sigma, \delta_1$ and $\delta_2$) are given in Figure 7. The only difference between these two $\mu$-charts is that the transition in the chart $C_1$ is triggered by the presence of the signal $a$ whereas the transition in chart $C_2$ is triggered by the absence of the signal $a$. Both examples output and feedback $a$.

![Fig. 7.](image)

The encoding of feedback is present in the schemas that describe the respective transitions in these charts.
Given $\Psi = \{a\}$, the respective predicates $a \in i_{C_1} \cup (o_{C_1} \cap \Psi)$ and $a \notin i_{C_2} \cup (o_{C_2} \cap \Psi)$ mean that both of the transitions’ guards rely not only on the environment’s input, e.g. $i_{C_1}$, but also on the fed back output, e.g. $(o_{C_1} \cap \{a\})$.

Now we can state and prove lemmas that describe exactly the meaning of the two examples.

Firstly, for the chart pictured in Figure 7(a), we show that the transition from state $A$ to $B$ happens regardless of the input from the environment. For simplicity we make the assumption that the charts are active.

**Lemma 121** Given $\delta_{C_1} = [(C_1, \Sigma, \sigma, \Psi, \delta_1)]_{Z_t}, \Sigma = \{A, B\}, \sigma = A, \Psi = \{a\}$ and $\delta_1 = \{(A, B, a/a)\}$, for arbitrary $z \in T$ and input $i \subseteq in_{C_1}$ we have,

\[
\text{active } C_1 z \quad z = \{ c_{C_1} \Rightarrow A, i_{C_1} \Rightarrow i, c'_{C_1} \Rightarrow B, o_{C_1} \Rightarrow \{a\} \}
\]

where $\llbracket \delta_{C_1} \rrbracket^T_T$

Hence, the meaning of the chart $C_1$ is that the transition from state $A$ to state $B$ always happens in the first step of the chart, that is regardless of the input that the environment offers, and the signal $a$ is output. This also implies that this chart has identical behaviour to a similar chart in which the transition is labeled $/a$, i.e. a true trigger.

Likewise, we show that the transition model for the chart $C_2$ pictured in Figure 7(b) does not contain any bindings that model the transition from state $A$ to $B$. This is because the single transition in chart $C_2$ is inconsistent in the presence of feedback on the signal $a$ under any input. Essentially, the output invalidates the trigger of the transition, hence the transition can never happen. This is expressed by the following lemma.

**Lemma 122** Given $\delta_{C_2} = [(C_2, \Sigma, \sigma, \Psi, \delta_2)]_{Z_t}, \Sigma = \{A, B\}, \sigma = A, \Psi = \{a\}$ and $\delta_2 = \{(A, B, a/a)\}$, for arbitrary $z \in T$,

\[
\text{active } C_2 z \\
\]

where $\llbracket \delta_{C_2} \rrbracket^T_T$
In fact in this simple example the transition model is the empty set of bindings because the inconsistent transition is the only candidate to be modeled. In general any inconsistent transitions can be removed from the chart without changing its behaviour.
The composition operator allows us to take two $\mu$-charts $C_1$ and $C_2$ and join them together to form a new more complex chart $C_1 \mid \Psi \mid C_2$ where $\Psi$ is a set of signals. As mentioned, we assume that the charts run separately but synchronously, i.e. in lock step with one another. Their only medium of communication is asynchronous via the multicast of signals. The set $\Psi$ denotes the signals that the charts $C_1$ and $C_2$ can communicate on. The communication is asynchronous in that output is always enabled, that is a chart can always broadcast signals, however there is no guarantee that the other chart in the composition is listening, in other words ready to react on the signals broadcast. Signals persist only during one step of the chart.

The following diagram demonstrates the structure of the composed chart $(C_1, \Sigma_1, \sigma_1, \Psi_1, \delta_1) \mid \Psi \mid (C_2, \Sigma_2, \sigma_2, \Psi_2, \delta_2)$.

Notice that any signals that the sets $\Psi$ and $\Psi_1$ (respectively $\Psi$ and $\Psi_2$) have in common will be fed back on two separate paths in this diagram. Also, the composition operator not only allows $C_1$ to communicate with $C_2$ using the signals in $\Psi$ but effectively changes the feedback characteristics of $C_1$ itself. The output that $C_1$ treats as input via feedback is now all of the signals in $\Psi_1 \cup \Psi$.

Both charts in the composition have an equivalent active state, i.e. one chart is active if and only if the other is active. In the circuit diagram we use dotted lines as a shorthand to indicate that the single active input to the composition is linked to both of the active inputs from the parts of the composition.

This example of the composition operator assumes that $C_1$ and $C_2$ are sequential charts. The operator in general however is defined over arbitrary charts. Hence the general picture for a composed chart $C_1 \mid \Psi \mid C_2$, where $C_1$ and $C_2$ are any charts, is as follows.
The transition model for composed charts is constructed by recursively constructing transition models for the two constituent parts of the composition. The base case being when the parts are themselves sequential charts. The model of the parts are then combined to create the transition model for the composition.

The transition model for the composed chart \( C = (C_1 \mid \Psi \mid C_2) \) contains the following \( Z \) definitions and schemas.
comes from the transition model of the chart \( C \) primarily, the bindings that inhabit the set \([\llbracket C \rrbracket]\). Arbitrary composed chart \( C \) composed charts directly in terms of the meaning of the Z model. Given an informal circuit diagrams.

The Z makes the obvious assumption that any entity subscripted with \( C_1 \) comes from the transition model of the chart \( C_1 \) and similarly for \( C_2 \). Importantly, the bindings that inhabit the set \([\delta_C] \) (i.e. the semantics of the schema \( \delta_C \)) have a similar signature to those in both \([\delta_{C_1}] \) and \([\delta_{C_2}] \). This shows that the Z model is consistent with the modular nature of charts demonstrated by the informal circuit diagrams.

As with sequential charts we give the definition of the transition model for composed charts directly in terms of the meaning of the Z model. Given an arbitrary composed chart \( C = C_1 \mid \Psi \mid C_2 \) we have,

\[
[\delta_C]^T =_{def} \{ z \in T \mid \exists z_1 \in \delta_{C_1}, \exists z_2 \in \delta_{C_2} \mid z = z_1 \land z = z_2 \land z_1, i_{C_1} ? = (z_1 ? \cup (z_2, o_{C_2}) \land in_{C_2} \land z_2, i_{C_2} ? = (z_1 ? \cup (z_2, o_{C_2}) \land in_{C_2} \land z, o_{C_1} ! = z_1, a_{C_1} ! \cup z_2, a_{C_2} ! \land active C_1 z \leftrightarrow active C_2 z) \}
\]

From this definition we derive the following introduction and elimination rules for composed charts.

**Proposition 131** Given \( C = C_1 \mid \Psi \mid C_2 \) and assuming the usual conditions for \( z_1, z_2 \) and \( Q \) we have,

\[
\frac{z \in \delta_C \quad z_1 \in \delta_{C_1}, z_2 \in \delta_{C_2}, \exists z_1, z = z_1, \exists z_2, P_1, P_2, P_3, P_4 \vdash Q}{Q} \quad (\mathsf{l-\neg})
\]

\[
\frac{z_1 \in \delta_{C_1}, z_2 \in \delta_{C_2}, \exists z_1, z = z_1, \exists z_2, P_1, P_2, P_3, P_4 \vdash Q}{z \in \delta_C} \quad (\mathsf{l-\neg})
\]

\[
\frac{z \in \delta_C \quad z_1 \in \delta_{C_1}, z_2 \in \delta_{C_2} \quad \exists z_1, z = z_1, \exists z_2, P_1, P_2, P_3, P_4 \vdash Q}{z \in \delta_C} \quad (\mathsf{l-\neg})
\]

26
where $P_1$ is $z_1. oc \sim (z. oc \cup \{\Psi\}) \cap inC_1$, $P_2$ is $z_2. oc \sim (z. oc \cup \{\Psi\}) \cap inC_2$, $P_3$ is $z. oc! = z_1. oc! \cup z_2. oc!$, and $P_4$ is active $C_1 z \iff$ active $C_2 z$

The predicates active $C z$ and inactive $C z$ are defined for composed charts as,

- active $C z =_{def} active C_1 z \lor active C_2 z$
- inactive $C z =_{def} inactive C_1 z \land inactive C_2 z$

While it may seem counter intuitive to have the disjunction in the definition of active $C z$, it is so defined in order that we have the property inactive $C z \iff \neg$(active $C z$) for composed charts.

We introduce a more useful form of the rule (indecreasing) as follows 8.

**Proposition 132** For arbitrary $o_1, o_2 \in P outC$,

\[
\begin{align*}
& z. oc! = o_1 \cup o_2 \\
& z \times \{ iact_2 \Rightarrow (z. ic? \cup fb z) \cap inC_1, oc_2! \Rightarrow o_1 \} \in \deltaC_1 \\
& z \times \{ iact_2 \Rightarrow (z. ic? \cup fb z) \cap inC_2, oc_2! \Rightarrow o_2 \} \in \deltaC_2 \\
& \text{active } C_1 z \iff \text{active } C_2 z \\
& z \in \deltaC
\end{align*}
\]

\[(\text{indecreasing})\]

where $fb z =_{def} z. oc! \cup \Psi$.

Also from (increasing) and (indecreasing), we can derive some other necessary introduction and elimination rules.

**Proposition 133** Given $C = C_1 \mid \Psi \mid C_2$, for arbitrary $z \in T$,

\[
\begin{align*}
& z \in \deltaC \quad \text{inactive } C z \quad \text{inactive } C z \\
& z \in \mathcal{Z} ChartC \\
& \text{inactive } C z \quad z \in \mathcal{Z} ChartC \\
& \text{inactive } C z \quad z \in \deltaC
\end{align*}
\]

\[\text{(iact\textsuperscript{−})} \quad \text{(iact\textsuperscript{−})} \quad \text{(iact\textsuperscript{+})}\]

where $[[\deltaC]]^P_T$.

Notice that the rules (iact\textsuperscript{−}), (iact\textsuperscript{−}) and (iact\textsuperscript{+}) have an identical form to the respective rules ($z_1\gamma_2$), ($z_2\gamma_3$) and ($z_3\gamma_4$) from Section 11. In fact, eventually we prove rules of this form for all chart regardless of their composition. Hence, we can assume they hold for all valid $\mu$-charts.

That the basic requirement of symmetry holds of the composition operator is given by the following rule 9.

**Proposition 134**

\[
\deltaC_1|\Psi|C_2 = \deltaC_2|\Psi|C_1
\]

\[\text{(iact\textsuperscript{−})}\]

Like for sequential charts (see Proposition 112), we can prove the rule ($z_3\gamma_2$) for composed charts.

---

8 The binding concatenation operator $\times$ is defined as in [2].

9 Recall the transition model $[\deltaC]$ is a set of bindings so the notion of equality used here is the usual set equality

27
Proposition 135: Given \( C = C_1 \mid \Psi \mid C_2 \),

\[
\forall z \in \delta_C \quad z_{i_c?} \cup fb z = x \quad \text{for some } x \in \delta_C
\]


\( (Z_{\xi}^c) \)

where \([\delta_c]^T, T_i = T - [c_{i,?} : \text{in}_c] \) and \( fb z = \omega(f, z, oc_c) \cap \Psi \).

The proof of Proposition 135, that is that the rule \((Z_{\xi}^c)\) holds for composed charts of the form \( C = C_1 \mid \Psi \mid C_2 \), relies only on the fact that the rule \((Z_{\xi}^c)\) itself holds for each part of the composition, i.e. \( C_1 \) and \( C_2 \). Hence, by an inductive argument, \((Z_{\xi}^c)\) holds regardless of the structure of \( C_1 \) and \( C_2 \).

As expected, given the instantaneous feedback of signals in charts, whenever the fed back output from one part of the composition, say \( C_1 \) triggers a transition in the other part, say \( C_2 \), there is a corresponding binding in the transition model of the composition. This binding represents a transition of the composition that is triggered by the same input as the transition in \( C_1 \) and outputs the combination of the outputs from the respective parts. That this is the case follows directly from Proposition 135.

We give a concrete example of using this rule. Consider the following \( \mu \)-chart.

Given this chart we can easily show that

1. \( \langle c_{C_1} \Rightarrow A, c'_{C_1} \Rightarrow B, i_{C_1}? \Rightarrow \{a\}, active_{\mu} \Rightarrow \{C_1, C_2\}, oc_{C_1}! \Rightarrow \{b\} \rangle \in \delta_{C_1} \)
2. \( \langle c_{C_2} \Rightarrow C, c'_{C_2} \Rightarrow D, i_{C_2}? \Rightarrow \{b\}, active_{\mu} \Rightarrow \{C_1, C_2\}, oc_{C_2}! \Rightarrow \{c\} \rangle \in \delta_{C_2} \)

From (1) and (2) by \((\bowtie_\bot^+)\) we have,

3. \( \langle c_{C_1} \Rightarrow A, c_{C_2} \Rightarrow C, c'_{C_1} \Rightarrow B, c'_{C_2} \Rightarrow D, i_{C_1}? \Rightarrow \{a, b\}, active_{\mu} \Rightarrow \{C_1, C_2\}, oc_{C_1}! \Rightarrow \{b, c\} \rangle \in \delta_C \)

Now from (3) and Proposition 135, given that \( \{a\} \cup \{b\} = \{a, b\} \cup \{b\} \), we can show that

4. \( \langle c_{C_1} \Rightarrow A, c_{C_2} \Rightarrow C, c'_{C_1} \Rightarrow B, c'_{C_2} \Rightarrow D, i_{C_1}? \Rightarrow \{a\}, active_{\mu} \Rightarrow \{C_1, C_2\}, oc_{C_1}! \Rightarrow \{b, c\} \rangle \in \delta_C \).

Also holds. This binding realizes the behaviour of the composed chart \( C \), that is, assuming initial states, when the environment offers the signal \( a \), both charts \( C_1 \) and \( C_2 \) make a transition. Clearly the transition in \( C_1 \) is triggered by the input \( a \). The transition in \( C_2 \) however, is triggered by the instantaneous feedback of the output signal \( b \).
This example demonstrates that the transition model essentially recursively sequentializes” composed charts. By “sequentialize” we refer to the process of taking the cross product of all of the transitions in each part of the composition and creating a transition in the composition model that represents both of these transitions happening together. This process is recursive whenever the parts of the composition are not sequential μ-charts. Consider the following illustration of this process.

\[\text{Fig. 8.}\]

The chart \(C_4\) of Figure 8(b) is the sequential representation of \(C_1 \mid \{a, b\} \mid C_2\). Chart \(C_5\) is the sequential representation of \(C_4 \mid \{a, b\} \mid C_3\).

That the following holds,

\[\begin{align*}
\&c_{C_1} \ni A, c_{C_2} \ni C, c_{C_3} \ni E, c'_{C_1} \ni B, c'_{C_2} \ni D, c_{C_3} \ni F, \\
\&ic?\ni\{a, b, c\}, active_{\underline{\underline{\delta}}} \ni \{C_1, C_2, C_3\}, oc!\ni\{b, c, d\} \ni \delta_C.
\end{align*}\]

and hence,

\[\begin{align*}
\&c_{C_1} \ni A, c_{C_2} \ni C, c_{C_3} \ni E, c'_{C_1} \ni B, c'_{C_2} \ni D, c_{C_3} \ni F, \\
\&ic?\ni\{a\}, active_{\underline{\underline{\delta}}} \ni \{C_1, C_2, C_3\}, oc!\ni\{b, c, d\} \ni \delta_C.
\end{align*}\]

where \(C = (C_1 \mid \{b, c\} \mid C_2) \mid \{b, c\} \mid C_3\), that is the chart pictured in Figure 8(a), is trivial to show using the same method as the previous example.
Of course, that this process works correctly, that is models our intuition for charts that contain negated signals, requires that the combination of inconsistent transitions is not represented in the transition model. Take for example the following chart.

The following lemma states, assuming that the charts $C_1$ and $C_2$ are active, the transition model for this chart contains no bindings that represent a transition from the configuration $A, C$ to the configuration $B, D$. Hence, even though there are explicitly defined transitions in charts $C_1$ and $C_2$ respectively, the result of combining these two charts using composition is that the composed chart has no explicitly defined transitions.

**Lemma 136** Given $C = C_1 | \{ b \} | C_2$, for arbitrary $z \in T$ we have,

$$\{ C_1, C_2 \} \subseteq z.active_{-}$$

where $| \delta_C |^T = T$

The fact that, in this particular example, we prove that there are no explicit transitions at all in the model of the composition, i.e. $| \delta_C | = \{ \},$ is because the example has only two candidate transitions that can combine to cause an explicit transition in the composition. Because these transitions are inconsistent, given the defined feedback, the resulting set of bindings that is the partial relations semantics for this composed chart is empty.
Another of the useful structuring mechanisms of $\mu$-charts is the decomposition operator. The decomposition of a sequential chart refers to replacing a state in the chart with another $\mu$-chart. This creates a master / slave relationship between the sequential chart, the master, and the arbitrary chart, the slave, that replaces a state in the master.

Intuitively, we can consider the behaviour of a master and a slave in such a chart as though they are composed in parallel. Consider the following circuit diagram representing the chart $Dec\ (Master, \Sigma_1, \sigma_1, \Psi_1, \delta_1)$ by $\{(\sigma_1, (Slave, \Sigma_2, \sigma_2, \Psi_2, \delta_2))\}$.

There are two subtle but significant differences between this diagram and the corresponding circuit diagram for the composition operator. The first being that the active state of the slave is determined by the master and not by the active input to the overall decomposition. This is because the state of the master determines whether or not the slave is active. The second difference is that the feedback signals that the master and slave share or communicate on are determined by the feedback set that is present in the definition of the master, i.e. denoted by the links label $\Psi$ in the diagram.

Hence when the master is in the state decomposed by the slave, that is both charts are active, then the decomposition is exactly the same as the composition of the master and slave with the feedback set equivalent to that of the master chart.

Again, the structure of the circuit diagram assumes both the master and slave are sequential charts. In general the slave can have arbitrary structure. However, unlike composition, the master must be a sequential chart.

The semantics that results from adhering exactly to this model of decomposition produces some “unexpected” behaviour (at least behaviour that may be considered somewhat counter to intuition). For example consider the decomposed chart $C = Dec\ (\omega \ C_1)$ by $\{(C_2, \omega C_2)\}$ pictured in Figure 9.
The semantics of the example decomposed chart is demonstrated by the composed chart $C' = \{a,c,e,g\} \mid C_1 \mid \{activeC2\} \mid C_2(b,d,f,h)$ pictured in Figure 10(a). The chart $C_3$ (Figure 10(b)) is the sequential equivalent of $C'$. Hence the chart $C_3$ exhibits identical behaviour to the original decomposed chart $C$.

The behaviour of the chart $C_3$ does not appear to capture the intention of the decomposed chart $C$ under any standard interpretation. However, consider the following examples of complimentary interpretations that we may wish to capture using the decomposition operator. Under some situations we may wish that the slave chart $C_2$ is initialized each time a transition in the master enters the slave. On the other hand we may wish that the slave remembers what state it was in when it was last exited by the master. It may be the case that the
master should be idle, that is remain in the slave’s state and produce no output, while the slave is free to make transitions or we may wish to allow the master to perform some task in parallel with the slave, for example counting the number of transitions the slave makes. Importantly, each of these interpretations can be implemented using explicit transitions in the original decomposed chart such that the new behavior is a refinement of the behaviour that is assigned to chart $C$. To make explicit in the semantic model one of these interpretations for the chart $C$ would have the side-effect that the alternate interpretation is no longer expressible at all. Hence by giving the most general semantics we allow different interpretations to be assigned in the future without modifying the language.

In general a clear goal of any semantic model is to capture our intuition. However, we choose to keep the semantic model of charts as simple and general as possible and modify our intuition slightly. We make this choice because, not surprisingly, it is often the case that the intuition that we have is based on the example uses that have been recently encountered (or at best is limited to the ability to imagine future uses). Hence, while the initial temptation is to complicate the model so that it exactly captures our current intuition, inevitably a new example arises which counters that intuition and therefore invalidates the complicated model. The ability to make good choices about the semantics highlights one of the benefits of a formal treatment of a language. When faced with several seemingly valid choices for meaning we can often rely on the eloquence and generality of the formal model to determine the most useful meaning.

The transition model for the decomposed chart $C$ such that $C = \text{Dec} \omega M$ by $(S, \omega S)$, where $\omega M = (M, \Sigma, \sigma, \Psi, \delta)$ and arbitrary $S$, contains the following Z definitions and schemas.

\[
\begin{align*}
\text{states}_C &= \mathbb{P}\mu_{\text{State}} \\
\text{in}_C &= \mathbb{P}\mu_{\text{Signal}} \\
\text{out}_C &= \mathbb{P}\mu_{\text{Signal}} \\
\text{states}_C &= \text{states}_M \cup \text{states}_S \\
\text{in}_C &= \text{in}_M \cup \text{in}_S \\
\text{out}_C &= \text{out}_M \cup \text{out}_S
\end{align*}
\]

\[
\begin{align*}
\delta_C \Delta \text{Chart}_C \\
i_C? : \mathbb{P}\text{in}_C \\
\text{active}_C : \mathbb{P}\mu_{\text{State}} \\
o_C! : \mathbb{P}\text{out}_C \\
\text{active}(S) \iff (\text{active}(M) \land (MS \lor MS')) \\
\exists i_M?, i_S?, o_M!, o_S! : \mathbb{P}\mu_{\text{Signal}} \cdot i_M? = (i_C? \cup (o_C! \cap \Psi)) \cap \text{in}_M \land \\
i_S? = (i_C? \cup (o_C! \cap \Psi)) \cap \text{in}_S \land o_C! = o_M! \cup o_S! \land \delta_M \land \delta_S
\end{align*}
\]
Again we give the definition of the transition model in terms of sets of bindings. Given the arbitrary decomposed chart \( C = \text{Dec} (\omega M) \) by \( \{(S, \omega S)\} \), where \( \omega M = (M, \Sigma, \sigma, \Psi, \delta) \) and arbitrary \( S \) we have,

\[
\deltaC^T = \text{def} \{ z \in T \mid \exists z_m \in z_m \in \deltaM; z_s \in \deltaS \cdot z_m = z_m \wedge z_s = z_s \wedge \left( z \cdot oC! = z_m \cdot oM! \cup z_s \cdot oS! \wedge z_m \cdot iM = (z \cdot iC? \cup (z \cdot oC! \cap \Psi)) \cap \text{inM} \wedge z_s \cdot is = (z \cdot iC? \cup (z \cdot oC! \cap \Psi)) \cap \text{inS} \wedge \right) \text{active } S \ z \iff \text{(active } M \ z \wedge \left( \inM = S \lor c_M' = S \right)\}.
\]

Then the introduction and elimination rules for decomposed charts are as follows. Given \( C = \text{Dec} (\omega M) \) by \( \{(S, \omega S)\} \), where \( \omega M = (M, \Sigma, \sigma, \Psi, \delta) \), and assuming the usual conditions for \( z_m, z_s \) and \( Q \) we have,

**Proposition 141**

\[
z \in \deltaC \quad z_m \in \deltaM, z_s \in \deltaS \quad z \cdot z_m = z_m, z \cdot z_s = z_s, P_1, P_2, P_3, P_4 \vdash Q \quad (M^+_S)
\]

where \( P_1 \) is \( z \cdot oC! = z_m \cdot oM! \cup z_s \cdot oS! \), \( P_2 \) is \( z_m \cdot iM = (z \cdot iC? \cup (z \cdot oC! \cap \Psi)) \cap \text{inM} \), \( P_3 \) is \( z_s \cdot is = (z \cdot iC? \cup (z \cdot oC! \cap \Psi)) \cap \text{inS} \) and \( P_4 \) is \( \text{active } S \ z \iff \text{(active } M \ z \wedge (z \cdot c_M = S \lor z \cdot c_M' = S)\}

The proofs of the introduction and elimination rules are trivial given the definition for decomposition.

The definition of the predicates \( \text{active } C \ z \) and \( \text{inactive } C \ z \) for decomposed charts is the same as for composed charts.

\[
\text{active } C \ z = \text{def } \text{active } M \ z \lor \text{active } S \ z \\
\text{inactive } C \ z = \text{def } \text{inactive } M \ z \land \text{inactive } S \ z
\]

As expected, given the definition of decomposed charts, we can show that two charts that share a master / slave relationship react in the same way as if they are composed in parallel whenever the master is in the decomposed state and vice versa. This gives us another useful introduction and elimination rule.

**Proposition 142** Given \( \deltaMz = [\text{Dec} (\omega M) \ by \\{(S, \omega S)\}]_{\mathbb{Z}} \), for arbitrary \( S, \omega M = (M, \Sigma, \sigma, \Psi, \delta) \), and \( \deltaM|S = [\omega M \mid \Psi \mid \omega S]_{\mathbb{Z}} \), then for arbitrary \( z \in T \),

\[
\frac{z \cdot c_M = S \lor z \cdot c_M' = S \quad z \in \deltaMz}{z \in \deltaM|S} \quad (M^+_S)
\]

where \([\deltaMz]^T\).

34
On the other hand, when the master is not in a decomposed state the slave contribute no output and does not change state, that is the slave makes no transition. Hence we derive further elimination rules as follows.

**Proposition 143** Given \(C = \text{Dec} (\omega M)\) by \(\{(S, \omega S)\}\), for arbitrary \(S\) and \(\omega M = (M, \Sigma, \sigma, \Psi, \delta)\) and \(z \in T\),

\[
\begin{align*}
\frac{z \in \delta_C \quad z.c.M \neq S \quad z.c.M' \neq S}{\text{inactive } S z} & \quad (M_{\delta C}^\ast) \\
\frac{z \in \delta_C \quad z.c.M \neq S \quad z.c.M' \neq S}{z \bullet \| i_M? \Rightarrow z.i.C? \cap i_M, o_M! \Rightarrow z.o.C! \} \in \delta_M} & \quad (M_{\delta C}^\ast)
\end{align*}
\]

We split the rule \((M_{\delta C}^\ast)\) into three more specific rules.

**Proposition 144** Given \(C = \text{Dec} (\omega M)\) by \(\{(S, \omega S)\}\), for arbitrary \(S\) and \(\omega M = (M, \Sigma, \sigma, \Psi, \delta)\). For arbitrary \(o_m, o_s \in \mathbb{P}M_C\) and \(z \in T\),

\[
\begin{align*}
\frac{z.o.C! = o_m \cup o_s \quad z.c.M = S}{z \bullet \| i_M? \Rightarrow (z.i.C? \cup i_M) \cap i_M, o_M! \Rightarrow o_m \} \in \delta_M} & \quad (M_{\delta C}^\ast)
\end{align*}
\]

**Proposition 145** Given \(C = \text{Dec} (\omega M)\) by \(\{(S, \omega S)\}\), for arbitrary \(S, \omega M = (M, \Sigma, \sigma, \Psi, \delta)\) and \(z \in T\),

\[
\begin{align*}
\frac{z \in \delta_C \quad \text{inactive } C z}{z \in \mathcal{Z} \text{ Chart}_C} & \quad (\text{iact}_C^-) \\
\frac{z \in \delta_C \quad \text{inactive } C z}{z.o.C! = \{\}} & \quad (\text{iact}_C^-)
\end{align*}
\]

where \([\delta_C]^P_T\).

The rules \((\text{iact}_C^-)\), \((\text{iact}_C^-)\) and \((\text{iact}^+)\) hold for decomposed charts.
Finally, we show that the transition model for decomposed charts, like sequential and composed charts, does not distinguish between the source of input, that is the rule \((z^x)\) holds for decomposed charts.

**Proposition 146** Given \(C = \text{Dec} (\omega M) \equiv \{(S, \omega S)\}\), for arbitrary \(S \) and \(\omega M = (M, \Sigma, \sigma, \Psi, \delta)\),

\[
\begin{align*}
  & z \in \delta_C \quad \text{and} \quad T_i = z \quad \text{and} \quad \forall fb \ z = x.iC? \cup fb x
  \quad x \in \delta_C
\end{align*}
\]

where \([\delta_C]^P_T, T_i = T - [i?: P \, \text{in} C]\) and \(fb z = z \circ C! \land \Psi\).

### 15 Chart Context and Signal Hiding

The final structuring mechanism of \(\mu\)-Charts is the interface operator. As we have already mentioned, the interface operator allows the designer to specify the assumed context for a chart. This operator allows two conceptually different types of behaviour to be specified. The first is signal filtering and signal hiding. The second is choosing not to output particular signals.

Signal filtering and signal hiding is typically used to internalize the communication between parts of a complex specification. That is, it may be a requirement that two sub-charts composed together communicate with each other privately. That is, the private communication between the parts can neither be observed or interfered with from outside of the composition. This type of behaviour can be modelled by specifying the internal communication between the sub-charts and then using the interface operator to localise the signals used for the internal communication.

Alternately a designer may wish to identify that the context can be controlled by some signal but choose not to output that signal. This is modelled using \(\mu\)-Charts by specifying an output interface that contains more signals than the natural output interface for a chart. It is not until we consider refinement of \(\mu\)-charts that the justification for this subtle distinction between the natural and explicit output interfaces of a chart becomes apparent. Given the finite state machine nature of charts, it is tempting to think that the natural output interface of a \(\mu\)-chart describes all of the signals that the chart can output regardless of what context it is placed in. In fact, the ability to specify an explicit interfaces for a chart allows a designer to specify both the assumed context of a chart plus the input output behaviour of the reactive system in that context. The refinement calculus that we define for charts then allows refinements that both refine the behaviour described as well as changing the assumed context of the chart.

The following diagram demonstrates the general structure of the model for a chart \(C = x [C_1]_Y\), that is, chart \(C_1\) with explicitly defined input interface \(X\) and output interface \(Y\).
The following $Z$ definitions and schemas provide the transition model for the chart $C = X [C_1] Y$.

$$\begin{align*}
\text{states}_C & : \mathbb{P} \mu \text{State} \\
\text{inc}_C & : \mathbb{P} \mu \text{Signal} \\
\text{out}_C & : \mathbb{P} \mu \text{Signal} \\
\text{states}_C & = \text{states}_{C_1} \\
\text{inc}_C & = X \\
\text{out}_C & = Y
\end{align*}$$

$$\Delta \text{Chart}_C \quad \exists \text{i}_{C_1}, \text{o}_{C_1} : \mathbb{P} \mu \text{Signal} \bullet \text{i}_{C_1}? = \text{i}_{C_1}? \land \text{o}_{C_1}! = \text{o}_{C_1}! \cap \text{out}_C \land \delta_{C_1}$$

Notice that apart from the obvious distinction between the schema $\delta_C$ and the existing $\delta_{C_1}$, i.e. the new predicate that restricts the output, there are some more subtle differences introduced using the $Z$ type system. In particular the input observation $\text{i}_C?$ has the type $\mathbb{P} \text{in}_C$ rather than $\mathbb{P} \text{in}_{C_1}$. When reasoning about the schema $\delta_C$ this observation gets normalised so that its type, as for the input observation $\text{i}_{C_1}?$, in $\delta_{C_1}$, is $\mathbb{P} \mu \text{Signal}$. However, the normalisation also adds the additional constraint $\text{i}_C? \subseteq \text{in}_C$ to the predicate part of the normalised version of the schema $\delta_C$. Thus restricting the set of binding from $\delta_{C_1}$ to those that take input from the new input interface for the chart $C$ and hence modeling the hiding of input.

Interestingly, the process of hiding input restricts the set of binding that is the model of the original chart, whereas, output hiding requires that we modify the bindings that inhabit the model of the original chart. When we restrict the input interface of a chart we are making the specified reactive system less reactive. That is, there are less signals that the environment can use to effect a reaction from the system in question. On the other hand, hiding output does not affect the reactivity of the specified system, rather it lessens the ability of system to affect its environment.

Once again, we give the definition of the transition model in terms of the meaning of the $Z$. Given the chart $C = X [C_1] Y$ we have,
\[ [\delta_C]^T_P = \{ z \in T \mid \exists z_1 \in \delta_{C_1} \bullet z = z_1 \land z.iC? \subseteq \text{in}_C \land z.iC? = z_1.iC? \land z.oC! = z_1.oC! \land \text{out}_C \} \]

Notice that the predicate \( z.iC? \subseteq \text{in}_C \) is not strictly necessary in the above definition because it follows from the fact that the binding \( z \) is in the type \( T \). However, we leave it explicit to clearly demonstrate that the set of bindings is appropriately restricted by the hiding of input.

The following introduction and elimination rules follow trivially from this definition. Given \( C =_X [C_1]_Y \) and assuming the usual conditions for \( z_1 \) and \( Q \) we have for arbitrary \( z \in T \),

**Proposition 151**

\[
\begin{align*}
\frac{z_1 \in \delta_{C_1} \quad z = z_1 \quad z.iC? = z_1.iC? \quad z.oC! = z_1.oC! \land \text{out}_C}{z \in \delta_C} & \quad (x|\uparrow) \\
\frac{z \in \delta_C \quad z_1 \in \delta_{C_1}, z = z_1, z.iC? = z_1.iC?, z.oC! = z_1.oC! \land \text{out}_C \vdash Q}{Q} & \quad (x|\downarrow)
\end{align*}
\]

The predicates \( \text{active } C \ z \) and \( \text{inactive } C \ z \) are for charts with hiding defined trivially as,

\[
\text{active } C \ z =_\text{def} \text{active } C_1 \ z \\
\text{inactive } C \ z =_\text{def} \text{inactive } C_1 \ z
\]

The rules \((\text{iact}_-^+), (\text{iact}_-^-)\) and \((\text{iact}_+^+)\) hold for charts that contain hiding.

**Proposition 152** Given \( C =_X [C_1]_Y \), for arbitrary \( z \in T \),

\[
\begin{array}{c}
\frac{z \in \delta_C \quad \text{inactive } C \ z \quad z \in \Xi \ \text{Chart}_C}{z \notin \Xi \ \text{Chart}_C} & \quad (\text{iact}_-^-) \\
\frac{z \in \delta_C \quad \text{inactive } C \ z \quad z.oC! = \{\}}{z \in \delta_C} & \quad (\text{iact}_-^+) \\
\frac{z \in \delta_C \quad \text{inactive } C \ z \quad z \notin \Xi \ \text{Chart}_C \quad z.oC! = \{\}}{z \notin \Xi \ \text{Chart}_C} & \quad (\text{iact}_+^+)
\end{array}
\]

where \([\delta_C]^T_P\).

And finally the rule \((Z^c)\) holds for charts with hiding.

**Proposition 153** Given \( C =_X [C_1]_Y \) and assuming that \( \Psi \) is defined to be the same as the feedback set \( \Psi_i \) for the chart \( C_1 \),

\[
\begin{align*}
\frac{z \in \delta_C \quad x | T_i = z \quad z.iC? \cup \text{fb } z = x.iC? \cup \text{fb } x \quad z \in \delta_C}{(Z^c)}
\end{align*}
\]

where \([\delta_C]^T_P\), \( T_i = T - [iC? : \text{P in}_C] \) and \( \text{fb } z =_\text{def} z.oC! \cap \Psi \).

38
16 Step semantics

In this section we give the general method for defining the step semantics for a chart.

The step semantics is no more than the transition model of the top-most subchart of a \( \mu \)-chart with the active state machinery hidden. Given an arbitrary \( \mu \)-chart called \( C \), the step behaviour of \( C \) is defined by another schema which, by convention, we call \( CSys \).

\[
\begin{array}{c}
\text{CSys} \\
\Delta Chart_C \\
i_C!: \mathbb{P} \text{ inc} \\
o_C!: \mathbb{P} \text{ out} \\
\exists \text{ active}: \mathbb{P} \mu_{\text{State}} \bullet \\
\text{active}(C) \land \\
\delta_C
\end{array}
\]

The schema \( CSys \) hides the active state observation and specifies that the topmost chart(s) in any hierarchical structure is (are) active.

Now the definition of the step semantics for a chart is as follows:

\[
\llbracket CSys \rrbracket^T =_{df} \{ z \in T \mid \exists z_1 \in \delta_C \bullet \text{active } C z_1 \land z = z_1 \}
\]

From this definition we derive the following introduction and elimination rules.

**Proposition 161** For arbitrary chart \( C \), assuming the usual conditions for \( z_1 \) and \( Q \) we have,

\[
\begin{array}{c}
\frac{z \in CSys \quad z_1 \in \delta_C, z = z_1, \text{active } C z_1 \vdash Q}{Q} \quad (Z^-) \\
\frac{z_1 \in \delta_C \quad z = z_1 \quad \text{active } C z_1 \quad z \in CSys}{(Z^+)}
\end{array}
\]

Now the schema \( CSys \) and its meaning describes the step semantics for \( \mu \)-Charts. We often refer to the step semantics as the *partial relations semantics*. This is because the meaning of the schema \( CSys \) can be considered as a relation that maps the before state of a chart and input to its after state and output. This relation is often partial, i.e. not defined for some configuration / input pairs, because an abstract \( \mu \)-Chart specification describes the reaction to some input events and not others.
References