

# **TECHNICAL REPORT**

**Department of Computer Science**

**A Graphical Representation for Communicating Sequential Processes**

**Maria Kutar, Carol Britton and Sara Jones**

**Report No: 314**

**June 1997**

## **A Graphical Representation for Communicating Sequential Processes**

**Maria Kutar, Carol Britton and Sara Jones**

**Department of Computer Science**

**University of Hertfordshire**

**College Lane, Hatfield, Herts, UK**

**AL10 9AB**

**Tel: 01707 285124 / 284354 / 284370**

**e-mail: M.S.1.Kutar, C.Britton, S.Jones@herts.ac.uk**

### **Abstract**

The use of formal notations has many advantages in the specification of interactive systems. However, validating specifications written using formal notations is generally difficult as these specifications are often hard to understand for those less familiar with such notations. Whilst it is widely acknowledged that users should be involved in validating requirements in order both to check that a specification says what they intended it to say, and to assist developers in refining requirements relating to the usability of the system, most users are not familiar with the formal notations used by software engineers and cannot realistically be expected to validate formal specifications in their raw state. One notation which has been found to be useful in specifying the behaviour of interactive systems is CSP (Communicating Sequential Processes). This paper presents a graphical representation of CSP which has been designed with the intention of making it easier for users to understand, and hence validate, specifications written in CSP. This graphical representation has been evaluated by a range of potential users, and refined in the light of their comments. The resulting representation is also analysed in terms of notational properties derived from Green's work on cognitive dimensions. Directions for further work are discussed.

## 1 INTRODUCTION

It is recognised that models constructed using mathematically based specification languages can provide an effective yet precise means of communication between software engineers and other stakeholders involved in the validation process [15]. Communicating Sequential Processes (CSP) [12] has been shown to be a suitable formal notation to model not only system behaviour but also the interaction between the user and the system, allowing for a clearer understanding of system requirements [11]. However, in its original, text-based form CSP can be difficult for the untrained user to understand.

It is accepted that user involvement in the requirements engineering process is beneficial; this can, however, be difficult if formal methods are used, and evidence suggests that graphical representations are more accessible to users who are untrained in a notation. In this paper a graphical representation of CSP is introduced in order to enhance understanding of the notation by the non-expert or untrained user. The aim is to enhance the accessibility of the notation without change to the semantics. Whilst a graphical representation of LOTOS, a very similar language to CSP, has been proposed elsewhere [23], this was designed specifically with the developer rather than the untrained user in mind, and therefore does not possess those properties which we believe enhance accessibility of the notation.

In section 2 we present the motives behind the provision of graphics, arguing that they can increase understanding, and in section 3 we go on to identify some specific properties of notations which contribute to ease of understanding. In section 4 a summary of the graphical representations of CSP operators is shown, followed by CSP solutions to the problem of the 'Dining Philosophers' in both traditional and graphical form. Section 5 presents the results of a small-scale survey in which our proposals for graphical representation of CSP were evaluated by a number of representative users. The representation is also analysed in terms of the notational properties outlined in section 3. Finally in section 6 we summarise and give our conclusions.

## 2 MOTIVES FOR THE PROVISION OF GRAPHICS:

### 2.1 Increasing Understanding:

Requirements engineering forms a cycle of knowledge acquisition, conceptual modelling and validation [14]. During knowledge acquisition relevant parts of the application domain are abstracted before being modelled. The process of user validation requires the user and developer to ensure that the system modelled is the same as that which the user requires.

For certain types of system, such as safety-critical systems, there are advantages in writing the specification in a formal notation (i.e. using a mathematical or logic-based notation, via which various properties of the system may be proved.) For the user, the combination of this abstraction and a formal notation with which he is unfamiliar results in a specification which he may find difficult to relate to his requirements. Whilst the provision of graphics is not a panacea [17] a graphical representation can

assist the user in understanding a formal notation. 'Graphical' here can be taken to mean 'not purely textual'.

One of the claims most commonly made in favour of graphical representation is that communication involving graphical information can take place over a much higher bandwidth than is possible using sequential text-based representations alone. This has several implications. It means that users should find it much easier to obtain some form of 'gestalt' or overview of the information from graphical representations than from text. Scanning for salient information should therefore also be easier [13].

Another set of related claims concerns the way in which graphical representations of information may map onto the users own internal or mental representations of the same information. The strong claim here is that external graphical representations may map directly onto internal mental images with which people can reason directly using well-advanced visual recognition and pattern-matching capabilities. For example Rumalhart *et al* [18] suggest that since relations among real-world, physical objects are often understood in visual or spatial terms it should be easier to derive a mental model of an unknown system structure from a graphical representation than from a textual one.

The status of these claims is admittedly unclear, and it is certain that there are individual differences in the extent to which people tend to 'visualise' information. Indeed, Stenning and Gurr [22] found that not only do individuals differ in their understanding of information, but that the efficacy of presentation of different types of information, and the ability to uncover different types of error, varies according to the media used. It does, however seem likely that for some users a graphical representation might prove to be useful in providing exactly the kind of 'cognitively transparent representation' which is needed for effective comprehension [20]. In addition there is a great deal of evidence to support the claim that graphical information is much more easily remembered than textual or verbal information [21]. A greater proportion of pictorial information is retained, with recall being significantly better for pictures than words, and the capacity for recognition of pictures being almost limitless when measured under appropriate conditions. These factors indicate that for the user who is attempting to understand an unfamiliar notation, the provision of graphics should both increase initial understanding and accelerate learning.

Formal methods encourage precise yet abstract specification. In addition to the fact that the production of a specification necessarily results in a degree of abstraction, many systems include abstract entities such as data or knowledge structures. During the development of such systems it is necessary to represent this abstract information, and if the user is to be able to assist in supporting the necessary design and development tasks, we need to find some way of making the abstract concrete. Scientists have long been familiar with the idea of representing abstract data in the concrete form of graphs, pie charts, bar charts and histograms. These concrete representations have proved useful in conveying an appropriate understanding of the data, allowing users to compare, contrast and infer in useful ways. Diagrammatic representations of software systems have long been used in design, development and teaching. It therefore seems reasonable to assume that such representations have, to some extent at least, been seen to be useful.

### 3 HOW CAN EASE OF UNDERSTANDING BE IMPROVED?

#### 3.1 Cognitive Dimensions

One of the most useful concepts in the evaluation of notations for different purposes is that of cognitive dimensions [7][8]. Cognitive dimensions are not a set of criteria which may be satisfied to various degrees by different notations; rather they are aspects of notations which may be important and useful in specific situations. Cognitive dimensions are tools for thinking about notations, rather than detailed guidelines. They are intended to support the evaluation of any type of information structure and can be applied to programming languages, musical scores or even telephone numbers [16]. The twelve dimensions described by Green aim to provide a broad-brush assessment of the information structures to which they are applied. They are outlined in table 1 below [16]. All of the dimensions are useful in the overall evaluation of a modelling notation, but those shown in bold are those considered to be most directly relevant to ease of understanding for novice readers [2].

*Table 1: Overview of Green's Cognitive Dimensions*

DIMENSION	INFORMAL DEFINITION
Viscosity	Resistance to change
<b>Hidden Dependencies</b>	Important links between entities are not visible
<b>Visibility and Side-by-Side-ability</b>	Ability to view components easily
<b>Diffuseness/Terseness</b>	Succinctness of language
<b>Closeness of Mapping</b>	Closeness of representation to domain
Progressive Evaluation	Effort required to meet a goal
Hard mental Operations	Operations that place a high demand on working memory
Imposed Guess-Ahead	Constraints on the order of doing things
<b>Secondary Notation</b>	Extra information in means other than program syntax
<b>Abstraction Gradient</b>	Types and availability of abstraction mechanisms
<b>Role-Expressiveness</b>	The purpose of a program component is readily inferred
<b>Consistency</b>	Similar semantics are expressed in similar syntactic forms

#### 3.2 Properties of Notations Which Contribute to Ease of Understanding

Building on the work of authors such as Green [6][7][8][9][10], Petre [17], and Sampson [19], Britton & Jones have identified the following properties that may contribute to ease of understanding of representations.

- 1 The number of different symbols in the notation
- 2 The degree of motivation of the symbols

- 3 The discriminability of the symbols
- 4 The extent to which the notation is perceptual or symbolic
- 5 The amount of structure inherent in the notation

(These are all discussed fully in [2])

### 3.3 A Word of Caution

As we have noted above, and many other writers have cautioned [1][3][5][24], graphics is not a panacea. In [17] Petre explores some of the limitations of pictorial and graphical media. She notes, *inter alia*, that ‘the success of a representation, graphical or textual, depends on whether it makes accessible the particular information the user needs - and on how well it copes with the different information requirements of the user’s various tasks.’ Additionally she argues that graphical readership is an acquired skill and that structure, relationships and relevance are not universally obvious. Perhaps most importantly, the point is made that ‘much of what contributes to the comprehensibility of a graphical representation isn’t part of the formal (programming) notation but a “secondary notation” of layout, typographic clues and graphical enhancement that is subject to individual skill.’

‘Secondary notation’ refers to valuable layout clues that are typically not part of the formal notation, but which may be used to exhibit relationships and structures that are not otherwise apparent. It is this linking of perceptual clues to the important information which can contribute to the understandability of a graphical representation. Consequently the ‘success’ of a representation is dependant on the skill of the person who creates it as well as the notation he uses.

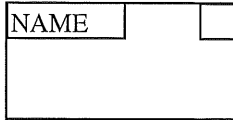
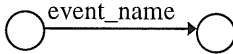

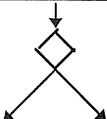
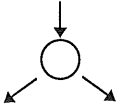

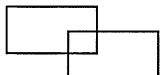
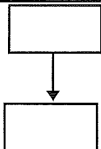
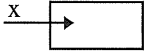
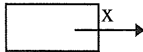

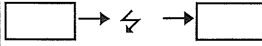
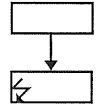
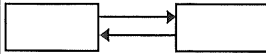
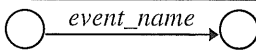
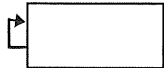
Finally, we should also be aware that whilst graphics have their limitations, they are, in Petres’ words, ‘nevertheless persistently appealing’ and this appeal may have its own value. The fact that a graphical representation is not necessarily a more coherent or understandable one is of less importance if the user believes it to be inherently more accessible.

## 4 GRAPHICAL CSP

### 4.1 Summary of graphical symbols for CSP

Table 2 outlines the proposed graphical symbols for the CSP operators. These symbols build on the methods employed by Hoare [12] in his explanation of CSP, and are designed with the intention that they are both intuitive and straightforward to learn. Thus Hoare’s diagrammatic representation of parallel processes is abandoned although that of events is retained. All other symbols are additional to those suggested by Hoare.

Table 2 Graphical CSP Symbols

OPERATOR	CSP SYMBOL	GRAPHICAL SYMBOL	NOTES
Process	Denoted by UPPER CASE		Process name will remain in upper case to follow the CSP convention. Labelling is included (see e.g.)
Event	Denoted by lower case		The circle shows the process waiting for an event to occur, the arrow the occurrence of the event
External choice (deterministic)			The diamond replaces the circle when waiting for environmental intervention
Internal choice (nondeterministic)			Denoted by divergent arrows
Parallel composition			Processes connected by parallel lines
Interleaving			Overlapping processes suggest interleaving
Sequential Composition	;		Control passes from top to bottom sequentially
Input	?		Named input shown into the process
Output	!		Named output shown from the process
Catastrophe			Lightning strike retained as it is an intuitive symbol
Interrupt	$P^{\wedge}Q$		Symbol chosen in accordance with survey results (see section 5.1)
Alternating	$\otimes$		The arrows show alternating transfer between processes
Hiding	$\backslash e$		Hidden events will be shown in italics
Restart	$\hat{P}$		The arrow indicates that the process may start again

#### 4.2 Illustration Of The Graphical Notation: *The Dining Philosophers*

In order to show the potential of this notation, Hoare's CSP solution to the problem of the dining philosophers [12] is shown both in the traditional CSP form and then graphically. The problem of the dining philosophers is first outlined.

In ancient times, a wealthy philanthropist endowed a College to accommodate five eminent philosophers. Each philosopher had a room in which he could engage in his professional activity of thinking; there was also a common dining room, furnished with a circular table, surrounded by five chairs, each labelled by the name of the philosopher who was to sit in it. The names of the philosophers were PHIL<sub>0</sub>, PHIL<sub>1</sub>, PHIL<sub>2</sub>, PHIL<sub>3</sub>, PHIL<sub>4</sub>, and they were disposed in this order anticlockwise around the table. To the left of each philosopher there was laid a golden fork, and in the centre stood a large bowl of spaghetti, which was constantly replenished.

A philosopher was expected to spend most of his time thinking; but when he felt hungry, he went to the dining room, sat down in his own chair, picked up his own fork on the left, and plunged it into the spaghetti. But such is the tangled nature of spaghetti that a second fork is required to carry it to the mouth. The philosopher therefore had also to pick up the fork on his right. When he was finished he would put down both his forks, get up from his chair, and continue thinking. Of course, a fork can be used by only one philosopher at a time. If the other philosopher wants it, he just has to wait until the fork is available again.

In traditional CSP form the representation of the dining philosophers is shown thus:

$$\alpha\text{PHIL}_i = \{i.\text{sits down}, i.\text{gets up}, i.\text{picks up fork.}i, i.\text{picks up fork.}(i\oplus 1), i.\text{puts down fork.}i, i.\text{puts down fork.}(i\oplus 1)\}$$

where  $\oplus$  is addition modulo 5, so  $i\oplus 1$  identifies the right hand neighbour of the  $i$ th philosopher. ( $\alpha\text{PHIL}$  defines the alphabet of the Philosopher process, showing the events in which he may engage.)

$$\alpha\text{FORK}_i = \{i.\text{picks up fork.}i, (i \ominus 1).\text{picks up fork.}i, i.\text{puts down fork.}i, (i \ominus 1).\text{puts down fork.}i\}$$

where  $\ominus$  denotes subtraction modulo 5.

Apart from thinking and eating, which we have chosen to ignore, the life of each philosopher is described as the repetition of a cycle of six events:

$$\text{PHIL}_i = (i.\text{sits down} \rightarrow i.\text{picks up fork.}i \rightarrow i.\text{picks up fork.}(i \oplus 1) \rightarrow i.\text{puts down fork.}i \rightarrow i.\text{puts down fork.}(i \oplus 1) \rightarrow i.\text{gets up} \rightarrow \text{PHIL}_i)$$

The role of the fork is a simple one - it is repeatedly picked up and put down by one of its adjacent philosophers (the same one on both occasions)



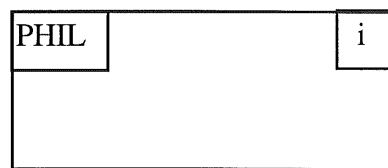
$$\text{FORK}_i = (i.\text{picks up fork.i} \rightarrow i.\text{puts down fork.i} \rightarrow \text{FORK}_i \\ | (i \ominus 1).\text{picks up fork.i} \rightarrow (i \ominus 1).\text{puts down fork.i} \rightarrow \text{FORK}_i )$$

Each event except sitting down and getting up requires the participation of two actors, a philosopher and a fork. The behaviour of the whole college is the concurrent combination of the behaviour of each of these components:

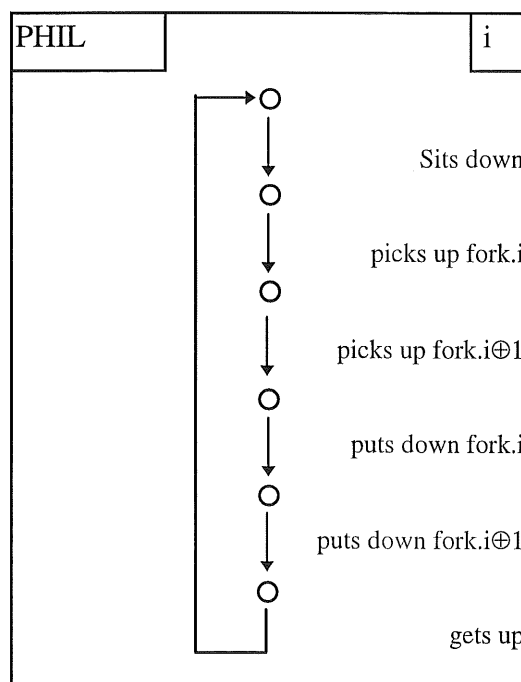
$$\text{PHILOS} = (\text{PHIL}_0 \parallel \text{PHIL}_1 \parallel \text{PHIL}_2 \parallel \text{PHIL}_3 \parallel \text{PHIL}_4 ) \\ \text{FORKS} = (\text{FORK}_0 \parallel \text{FORK}_1 \parallel \text{FORK}_2 \parallel \text{FORK}_3 \parallel \text{FORK}_4 ) \\ \text{COLLEGE} = (\text{PHILOS} \parallel \text{FORKS})$$

To a user unfamiliar with the CSP notation this is not particularly easy to understand. However, once the CSP is shown in a graphical form, without change to the CSP itself, things become a little clearer:

The PHIL process is shown as follows:



The inner box in the top left-hand corner shows the name of the process, that in the top right corner, the label which denotes the specific PHIL. By labelling the process in this way the events may be shown without their labels thus:

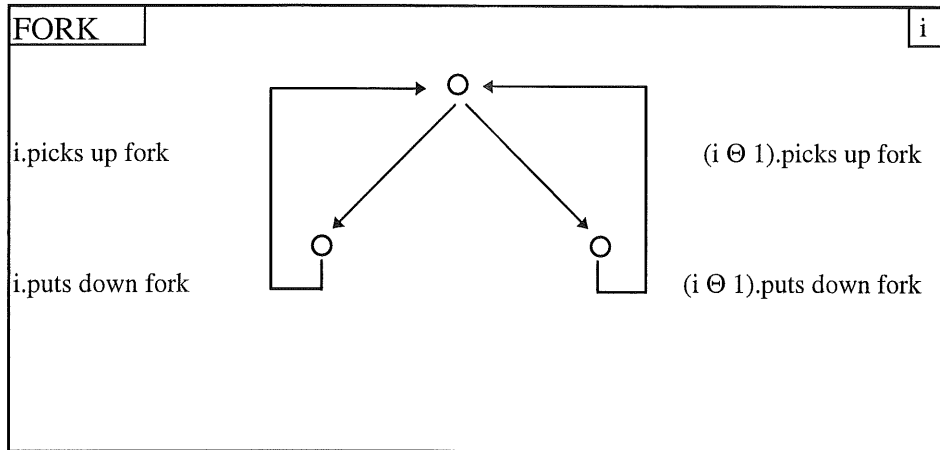


Whilst the inclusion of the label to specify the fork which the philosopher will use is still necessary, the removal of the label referring to the philosopher himself improves initial readability. The inclusion of arrows makes it clear that the events must follow

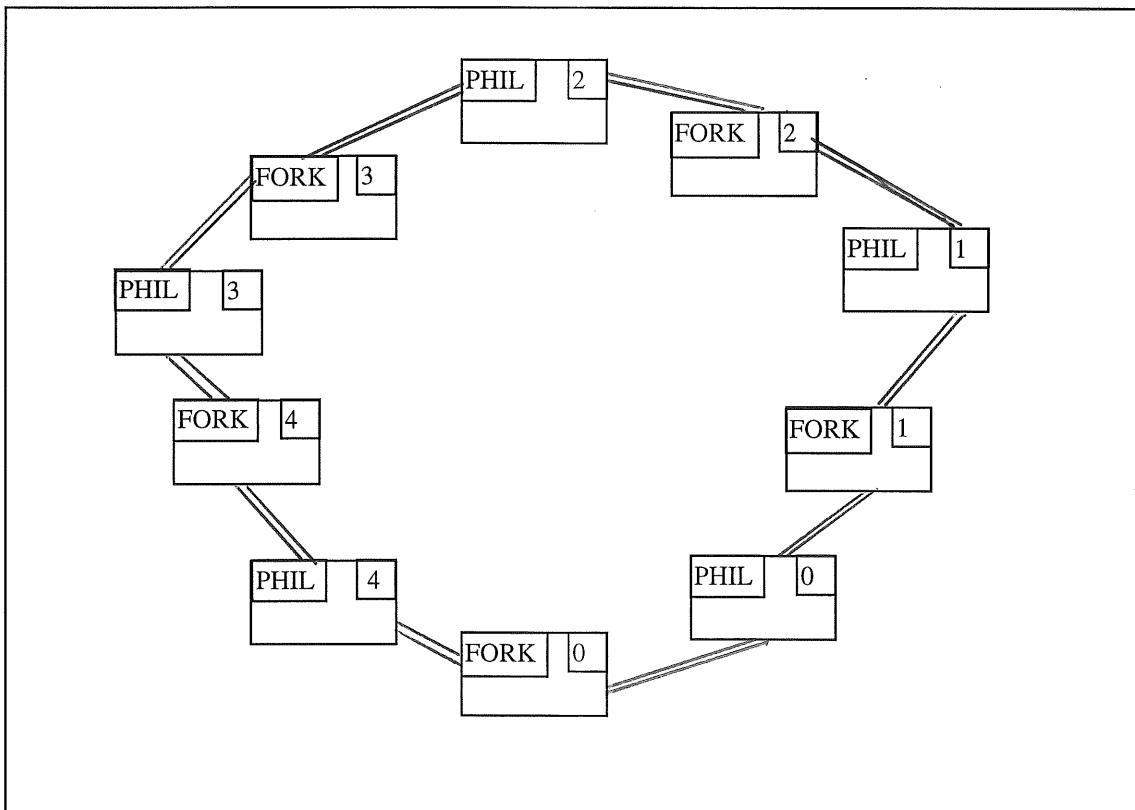
each other in succession and that the philosopher will engage in a repetition of this cycle of events.

The FORK process which offers the possibility of choice is also made clearer with the pictorial notation.

$$\text{FORK}_i = (i.\text{picks up fork}.i \rightarrow i.\text{puts down fork}.i \rightarrow \text{FORK}_i \\ | (i \ominus 1).\text{picks up fork}.i \rightarrow (i \ominus 1).\text{puts down fork}.i \rightarrow \text{FORK}_i)$$

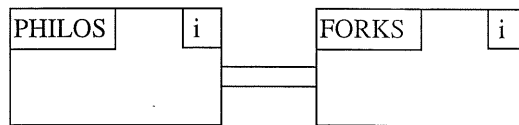


Then we may show the processes themselves running in parallel:

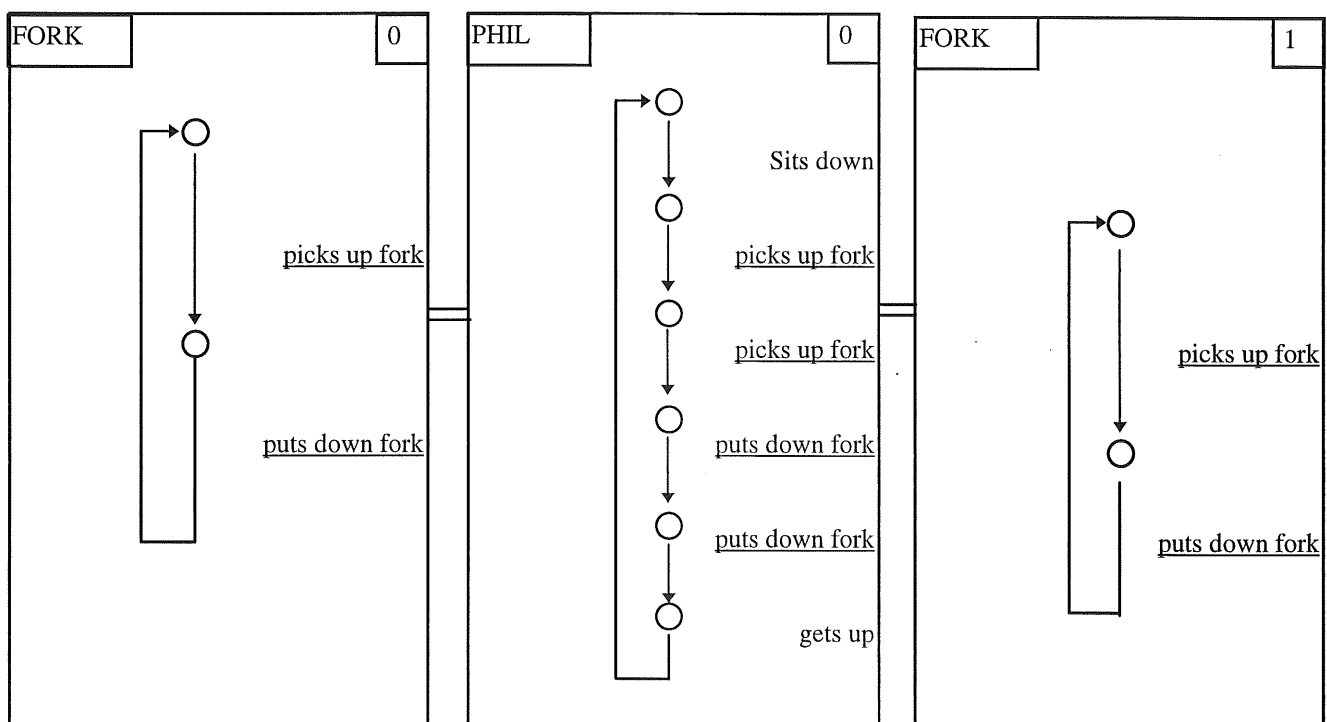


The use of secondary notation, showing the philosophers and forks in the order they will appear around the table aids understanding. However, it is possible to show the processes simply joined together as in

COLLEGE = PHILOS || FORKS



As was discussed in section 3.3 it is the use of secondary notation which aids understandability of graphics. Below, the processes  $PHIL_0$ ,  $FORK_0$ , and  $FORK_1$  are shown in parallel. The use of secondary notation allows a clear view of the events on which the processes must synchronise. We have underlined the events on which synchronisation is required.



The labelling of the processes allows the event names to be shown entirely without labels, and it can be seen that, for example, on the first 'picks up fork' event,  $PHIL_0$  picks up  $FORK_0$ . The processes are shown to be running in parallel and therefore it is apparent that  $PHIL_0$  and  $FORK_0$  must synchronise on this event.

## 5 EVALUATION OF GRAPHICAL CSP

In this section we present the results of a small survey designed to assess the ability of this notation to be understood by the untrained user. We then go on to consider the properties of this notation which contribute to ease of understanding.



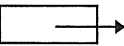
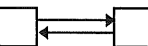
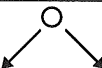
### 5.1 Empirical Evaluation

#### 5.1.1 Survey Details

To investigate whether this representation may be easily understood by the untrained user we have conducted a small-scale empirical study involving two groups of subjects who were broadly representative of the kind of users who might be involved in systems validation. The first group, of ten subjects, consisted of employees of Nortel. This group consisted of employees ranging from secretarial staff to senior managers. The second group was made up of five postgraduate students from the University of Hertfordshire. Some subjects from both groups had experience of formal notations. None had experience of CSP. Subjects were asked to complete a questionnaire without consultation with other members of the group and all questionnaires were completed anonymously. The questionnaire was devised to enable us to assess whether the graphical representations proposed in section 4 are intuitive to the untrained user. It consisted of nine multiple choice questions asking for a symbol to be selected which the subjects felt best represented the given word or phrase, as in the example shown below. Details of the results of the survey and some of the questions asked are included below.

*Example of question asked of subjects.*

You will see below a list of words and phrases used in everyday conversation. For each of these, please circle the letter corresponding to the symbol that you think best represents it.

6	output	A	B	C	D	E
						

#### 5.1.2 Survey Results

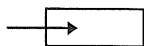
The choices presented in the questionnaire were made up of the symbol which we had designed to represent the CSP operator, any symbols which we felt might be confused with our choice, and a random selection of the remaining symbols. Representations of processes and events themselves were not included and we did not differentiate between internal and external choice. In both cases this was in order to circumvent the necessity of explaining the underlying CSP concepts to our subjects.

The following symbols which we had designed were selected by all fifteen subjects (100%)

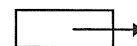
choice :



input:

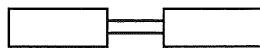


output:

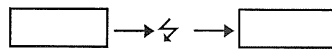


These symbols were selected by twelve or more of the subjects (80% + )

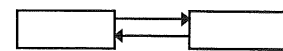
Parallel Composition:



Catastrophe:



Alternating:



The symbol for sequential composition:



was selected by 10 subjects (66%) and was retained.

The two remaining symbols which were included in the survey, were selected by none of the subjects. These were consequently altered to take into account the subjects' answers as detailed below.

Our question regarding interleaving was as follows:

4	interleaving	A	B	C	D	E

Our choice of symbol had initially been (A), based on the traditional CSP notation itself. However, fourteen of the subjects (93%) selected (C), our symbol for 'interrupt'.

The question relating to the interrupt symbol was posed as follows:

8	interrupt	A	B	C	D	E

Twelve subjects (80%) selected (D) , the symbol for catastrophe, and none chose our 'suggested' symbol (B). Discussion with some of the subjects revealed that (D) was chosen because they felt that the lightning strike symbolised an interrupt.

As a consequence of these results, we changed the interleaving symbol to that which was selected by the subjects, and designed a new symbol for interrupt incorporating a lightning strike.

The new symbol was created with the consideration that two symbols incorporating a lightning strike could be easily confused. The CSP definitions for the two operators are as follows.

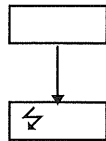
Catastrophe:

$P \bowtie Q$  behaves initially as process P. On the occurrence of the catastrophic event only, control will pass to Q.

Interrupt:

$P \hat{\bowtie} Q$  behaves initially as process P. On the occurrence of the first event from the alphabet of Q, control passes to Q and P is never resumed.

Thus we designed the following symbol for interrupt:



The lightning strike inside the process indicates that the second process may interrupt the first on the occurrence of the initial event *within* its alphabet. Keeping the lightning strike outside the process in the symbol for catastrophe indicates that the catastrophic event is required in order for control to pass between the two processes.

## 5.2 Theoretical Analysis of the Notation's Properties

In section 3.2 above we defined the properties of notations which contribute to ease of understanding. In this section we show that the graphical CSP representation possesses many of these properties.

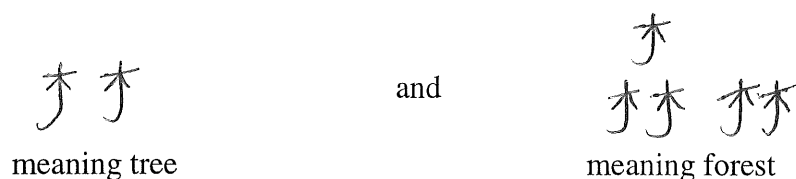
### 5.2.1 The number of different symbols in the notation.

This property is constrained by the original CSP. We have defined symbols for fourteen CSP operators, but many of these build on the symbol for a process using motivated symbols such as arrows (see below). Thus the overall number of new symbols which a user must learn is minimised. Some of the operators which in the original version of CSP are represented by symbols introduced specifically for this purpose (e.g. the ';' which is introduced to denote sequential composition, the '?' to denote input, and the '!' denoting output) are represented in our graphical version simply by new arrangements of a smaller set of symbols. We believe that this will tend to improve understandability so long as it does not lead to problems with discriminability (see section 5.2.3 below).

### 5.2.2 The degree of motivation of the symbols.

This property is perhaps the least self-explanatory but it is related to the cognitive dimensions of closeness of mapping and role expressiveness. The concept of

motivated and arbitrary notations has been described by Sampson [19]. A notation may be considered to be motivated if there exists a natural relationship between the elements of the notation and objects or ideas that they represent. Many of the characters in Chinese script are motivated, for example:



Not only does the form of the symbols suggest trees, but the symbols also preserve the idea that a forest is made up of many trees.

In an arbitrary notation there is no natural relationship between the object and the representation; the symbol \*, for example, bears no obvious relation to the notion of multiplication and the logic symbol  $\neg$  does not suggest negation.

While intuitively we may guess that symbols used in traditional CSP such as ‘?’ and ‘!’ denoting input and output, and ‘ $\otimes$ ’ denoting alternation will not be seen as natural by those not familiar with formal notations, the results of our survey suggest that at least some of the alternatives provided in our graphical version are seen as natural representations of the concepts they represent. For example the symbols representing input and output appear to be strongly suggestive of the ideas they represent and were selected by 100% of the subjects.



In designing these graphical symbols for CSP, motivation was a key priority. The survey results shown in section 5.1 indicate that the symbols we have chosen are indeed strongly motivated and it is hoped that this will assist the untrained user in understanding the notation.

### 5.2.3 The discriminability of the symbols.

This refers to the ease with which the symbols may be distinguished from each other. The nature of the CSP operators, showing mainly the relationship between processes tends to reduce the discriminability of the symbols. The fact that our graphical notation often represents new concepts by using new arrangements of existing symbols rather than by introducing new symbols, as described above, may also tend to reduce a readers’ ability to discriminate between the representations of different concepts. However, if representations are highly motivated, the effects of any lack of discriminability may be less apparent. Further work is needed to investigate whether the representations used in our graphical version of CSP are sufficiently discriminable.

#### **5.2.4 The extent to which the notation is perceptual or symbolic.**

Perceptual representations are those in which we perceive meaning directly without having to reason about them. Most modelling notations contain both perceptual and symbolic elements. Both the original and graphical versions of CSP are both perceptual and symbolic to a certain extent. The original version, whilst predominantly textual, may still use perceptual cues in layout and formatting of a specification; the graphical version, while having a strong perceptual flavour due to its reliance on graphical symbols still uses text, for example to name processes and events. We believe that the greater use of perceptual features will allow readers to benefit from the advantages of graphical representations outlined in section 2, however, further studies are needed to determine the extent to which this will affect an untrained users' ability to understand a specification.

#### **5.2.5 The amount of structure inherent in the notation.**

The design of the graphical CSP symbols incorporates a certain amount of structure, encouraging structurally clear representations. This is largely due to the increased scope for the use of secondary notation (see section 3.3). In the example of the Dining Philosophers above, we believe that greater clarity of structure has been achieved with the graphical version of CSP through careful layout of the processes with thought being given to the relationships which will be implied by the structure. Thus graphical CSP specifications are not inherently more structured than textual versions, but the graphical notation does at least provide the specifier with the possibility of making the structure of a specification clearer to its readers.

## **6 SUMMARY AND CONCLUSIONS**

It is accepted that formal specifications provide a precise method of communication between developers and users, and that effective systems development requires that the user of the system is able to validate requirements at an early stage. Given that users are generally unfamiliar with formal notations which a developer may use, this is a difficult task, particularly in light of the fact that modelling requires a certain amount of abstraction away from the system. Whilst the provision of graphics does not in itself provide a solution to this difficulty we have argued that they can both improve understanding and increase retention of information for some users.

Those properties of a notation which we believe to be directly relevant to the ease of understanding have been identified with reference to Green's cognitive dimensions. An important contributory factor for comprehension, additional to the properties of the notation itself, is the effective use of secondary notation. If a graphical representation is badly laid out, with little thought given to the relationships suggested by, for example, the grouping of objects in a diagram the reader is likely to draw incorrect conclusions. Thus graphical notations must be carefully used if they are to promote rather than hinder understanding.

CSP is a notation which may be used to model both system behaviour and the user interface and therefore it is a particularly suitable notation for the development of



interactive systems. We have introduced a graphical representation of CSP, which allows for the graphical depiction of a CSP specification without any changes to the semantics of the notation. Its potential has been illustrated through a worked example. Subsequently we have conducted a small-scale survey to assess the ease of understanding of this representation by the untrained user. The results suggest that the symbols we have chosen are intuitive. The properties of this representation which contribute to improved understanding have been discussed, although the fact that we are working with an existing notation means that we have been constrained in our ability to fully satisfy those properties. In order to fully assess the effectiveness of the graphical representation further studies need to be carried out to compare the novice users' understanding of the two different forms of CSP.

The representation is currently only in a paper based form, but it is hoped that it will provide a basis for a dynamic form which would allow users and developers to explore the interaction between processes which are used in CSP to represent the dynamics of the system. This could potentially be used as a graphical front end for existing CSP tools, such as FDR or ProBE [4] available for refinement and verification. This should help to overcome the difficulty of 'scaling-up' inherent in graphical notations and would be beneficial for both the user and the developer. Whilst the work has been carried out with specific regard for the user, systems developers may also find the graphical representation beneficial.

### **Acknowledgements**

The authors would like to thank Ben Potter for his advice and assistance in the development of this representation.

### **References**

- [1] Ambler,A., Burnett,M.: *Influence of Visual Technology on the Evolution of Language Environments*. IEEE Computer, 22, October 1989
- [2] Britton,C., Jones,S.: *The Untrained Eye: What Makes A Representation Easy For Novice Readers To Understand?* University of Hertfordshire Technical Report no 308 1997
- [3] Chang,S.: *Visual Languages: A Tutorial And Survey*. IEEE Software January 1987
- [4] CSP Archive: <http://www.comlab.ox.ac.uk/archive/csp.html>
- [5] Fitter,M.J., Green, T.R.G.: *When Do Diagrams Make Good Computer Languages?* In Alty,J. L., Coombs, M.J., (Eds) *Computing Skills and the User Interface*. Academic Press 1981
- [6] Green, T.R.G. *Programming as a Cognitive Activity*. In Smith, H.T., Green, T.R.G. (Eds) *Human Interaction with Computers*. Academic Press 1980
- [7] Green, T.R.G.: *Cognitive Dimensions of Notations*. People and Computers (HCI 89) Sutcliffe and McCauley (Eds) CUP 1989

- [8] Green, T.R.G.: *Describing Information Artefacts with Cognitive Dimensions and Structure Maps*. People and Computers, Proceedings of the HCI'91 Conference, Diaper, D., Hammonds, N. (Eds) August 1991
- [9] Green, T.R.G., Petre, M., Bellamy, R.: *Comprehensibility of visual and textual programs: A Test of Superlativism against the Match Mismatch Conjecture*. In Koenemann-Belliveau, J., Moher, T., Robertson, S. (Eds) *Empirical Studies of Programmers Fourth Workshop*, Norwood, NJ. 1991 Ablex pp121-146
- [10] Green, T.R.G., Blackwell, A.F.: *Thinking About Visual Programs*. In Thinking With Diagrams IEE Colloquium Digest No. 96/010. 1996
- [11] Harrison, M., Barnard, P.: *On Defining Requirements for Interaction*. In Proceedings of RE93, Second International Symposium on Requirements Engineering. IEEE Computer Society Press 1993
- [12] Hoare, C.A.R.: *Communicating Sequential Processes*. Prentice-Hall International 1985
- [13] Jones, S.: *Three-Dimensional Interactive Connection Diagrams for Knowledge Engineering*. PhD thesis, City University 1993
- [14] Lalioti, V., Loucopoulos, P.: *Visualisation for Validation*. CAISE'93, 5<sup>th</sup> International Conference on Advanced Information Systems Engineering, Paris, France, June 1993. In Rolland, C., Bodart, F., Couvet, C. (Eds) *Lecture Notes in Computer Science* 685.
- [15] Miller, S.P., Srivas, M.: *Formal Verification of the AAMP5 Microprocessor: A Case Study in the Industrial Use of Formal Methods*. In Proceedings of the Workshop on Industrial-Strength Formal Specification Techniques. IEEE Computer Society 1995
- [16] Modugno, F., Green, T.R.G., Myers, B.A.: *Visual Programming in a Visual Domain: A Case Study of Cognitive Dimensions*. In People and Computers IX, Proceedings of HCI'94, Glasgow, August 1994
- [17] Petre, M.: *Why Looking Isn't Always Seeing. Readership Skills and Graphical Programming*. Communications of the ACM, June 1995, Vol. 38, No. 6.
- [18] Rumelhart, D.E., Smolensky, P., McClelland, J.L., Hinton, G.E.: *Schemata and Sequential Thought Processes in PDP Models*. In McClelland, J.L., Rumelhart, D.E., and the PDP Research Group (Eds) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 2: Psychological and Biological Models*. MIT Press 1986
- [19] Sampson, G. : *Writing Systems*. Hutchinson 1985

[20] Seely Brown, J.: *From Cognitive to Social Ergonomics and Beyond*. In Norman, D., Draper, S. (Eds) *User Centred System Design*, Chapter 22, Lawrence Erlbaum Associates 1986

[21] Standing, L. *Learning 10,000 Pictures*. Quarterly Journal of Experimental Psychology . Vol. 25 1973

[22] Stenning, K., Gurr, C.: *Formal Methods and Human Communication*. In *Formal Aspects of the Human Computer Interface BCS-FACS Workshop*. Sheffield Hallam University 1996

[23] Winstanley, A.C., Bustard, D.W.: *Expose: an Animation Tool for Process-Oriented Specifications*. Software Engineering Journal, November 1991 pp 463-475.

[24] Wolf, C. (organiser) *The Role of Laboratory Experiments in HCI: Help, Hindrance or Ho-Hum?* In *Proceedings CHI89 1989*. (Panel Session)