New directions in Workflow formalisms

Frank Terpstra, Pieter Adriaans
Informatics Institute
University of Amsterdam
Kruislaan 419, 1098VA, Amsterdam, the Netherlands
{ftrpstra, pietera}@science.uva.nl

Abstract

Due to increasing complexity of scientific workflows as well as a greater number of available services new kinds of problems arise. In this paper we present an approach to these problems at multiple levels of both data communication and process abstraction. We give an overview of formalisms: Turing machines, I/O Automata, Petri Nets, Constraint Automata and π Calculus in respect of their suitability to the presented approach. On the basis of this overview we conclude that Constraint Automata best suit this approach.

1 Introduction

Within e-science research, workflow environments are are often viewed as a practical tool for building experiments. Right from the inception of these systems, formal foundations for workflows have been constructed [5, 9]. These first formal foundations provided the underpinnings for data flow based execution models and were based on for instance Kahn Process Networks [9]. For cases where more expressiveness was needed, for instance in hierarchical composition or more advanced control flow, scientific workflow management systems have turned to Petri Nets, see for instance [4]. Similarly in the business workflow community [3] where there has been more emphasis on expressiveness from the outset, the most commonly used formalism is Petri Nets. With the increasing complexity of workflows and the increasing abundance of available services to be used in their construction, new issues have arrived which need a formal basis as well. There are two very important criteria in scientific experiments, correctness and reproducibility, the formal foundations of a workflow environment should thus be able to prove that these criteria are met in an experiment. The ever expanding amount of web-services means that determining whether a web-service is suitable and correct for use in an e-Science experiment should be automated as much as possible. Not only is checking the type and semantics of a services connections needed, the runtime behaviour also has to be correct within the workflow its used in. The increasing complexity of workflows, for instance using multiple execution models within one experiment [12, 30] also increases the complexity of assuring reproducibility. The question is how much provenance data needs to be recorded and how this has to be coordinated between different workflow systems. These problems involve more than just the expressivity allowed by the execution model, and call for their own formalised description. This may involve additional and different formalisms than the formalisms now common in workflow research. These problems need to be defined in the formalism that is best suited to express this problem. In this paper we investigate existing formalisms both from workflow research as well as formalisms originating from parallel computing and software engineering. It is important to keep in mind that these formalisms are not workflow languages they are meant to represent existing workflow systems and languages at a more abstract level.

2 Problem domains

In e-Science, workflows are not just describing existing experiment processes, as business workflows describe existing processes. They are meant to explore new experiment ideas, scientific workflows are more constrained and need to be excactly reproducible. The design of these experiment workflows can be top down, bottom up or a combination of both. In a top down approach one starts with a very abstract design definition and gradually through refinement add more details. With a bottom up approach one starts with the atomic building blocks and gradually create a more abstract design by merging building blocks into compositions. During this design process abstraction and refinement of the workflow design play a crucial role. Within workflow systems such as ICENI [8] or Chimera [7] automated abstract to concrete workflow design is possible. Many work-

flow systems offer hierarchical composition where an entire workflow can be a workflow component, or sub-workflow, of another workflow. This concept becomes more interesting when a sub-workflow uses a different model of computation to that of the main workflow. It can be taken one step further by having a workflow component which is actually a workflow in a different workflow management system. The workflow-bus being developed within our group is an example of this [30]. Both Triana [15] and shortly Taverna [19], offer the option of exposing a workflow as a web-service. This allows a different workflow management system to use this web-service as a workflow component it its own workflow. Another problem which can be analysed using formal methods is that of connectivity. To determine which workflow components can be connected to form a valid workflow. This problem has two parts, first both data type and data semantics must match. Second there is the runtime aspect, are two components compatible at the connection level when they are executing? This second part comes to the foreground when dealing with multiple models of computation within one workflow. Whether runtime behaviour which is allowed under one model of computation is acceptable to a component operating under another model of computation, needs verification. This is also important in the reuse of workflow components. To test if a workflow that has been used for one experiment with certain data can be reused in another experiment which tries to find answers to a different problem. Finally, within science and therefore within e-science reproducibility of an experiment is very important. To ensure an experiment is reproducible provenance data is recorded during the execution of an experiment. When dealing with an experiment using multiple workflow management systems, the question is what provenance data needs to be shared between systems to maintain reproducibility.

3 Modelling workflow problems requiring abstraction

The problem to which formal methods are applied most often in workflow research is that of expressivity. To determine what control flow constructs are needed to define all desired workflows as well as which execution model and workflow language enables these control flow constructs. The problems defined in the previous section do not just deal with expressivity they deal with well formedness of the workflow. In this section we will describe a workflow design process at multiple levels of abstraction and show how this can help the problems mentioned in the previous section. In previous work [23] we defined a lattice for workflow component design , as illustrated by Figure1. This lattice deals with the amount of abstraction present in both the representation of process as well as communication. The

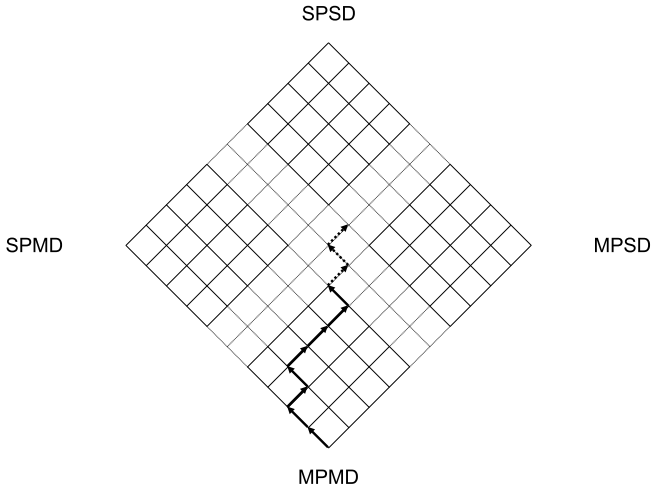

Figure 1. Lattice defining workflow design

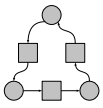

Figure 2. MPMD workflow, squares are processes circles are data connectors

Figure 3. MPSD workflow, squares are processes circles are data connectors

lattice is set up by the four most extreme cases:

- MPMD Multiple Process Multiple Data
- MPSD Multiple Process Single Data
- SPMD Single Process Multiple Data
- SPSD Single Process Single Data

Where Single process and Single data are the most abstract representations of process and data, while multiple process and multiple data are the most concrete or atomic representations. By either abstracting or refining data or process one can reach any point in this lattice and thereby any possible representation of the same computationally equivalent process. If Figure 2 is a MPMD representation with atomic processes and data connections, Figure 3 is its MPSD representation where all communication has been abstracted to one component. In Figure 4 the SPMD is shown, with all processes abstracted to one component. Finally Figure 5 shows the representation where both data communication and process are at maximum abstraction.

Figure 4. SPMD workflow, squares are processes circles are data connectors

Figure 5. SPSD workflow, squares are processes circles are data connectors

3.1 experiment representation

Now we will now more precisely define how workflows should be represented at different abstraction levels.

Problem definition: This is the most abstract representation of a workflow, it consists of a definition of desired input, as well as the desired output. These can either be defined in terms of data or processes. In e-science context it can also be the research question associated with an experiment.

Atomic workflow component: Represents a computational process for which there is a direct mapping to an actual implementation. This can be a deterministic process like adding two variables, or it can be a non deterministic process, a user entering a value based on the input he sees. In practise this usually is a web service, but it can also be an action performed by a human.

Execution model: The workflow component is modelled in the context of a workflow system by explicitly representing control flow. This can be done by modelling all data connectors allowed by an execution model. Workflow components should not contain anything specific to one workflow execution model.

Compositionality: is a property which can hold for workflow components, data connectors and a combination of both. Two or more atomic workflow components can be composed together to form a single composite workflow component. This composite component can hide its internal actions presenting itself to the outside only in terms of its inputs and outputs. Through compositionality complicated controlflow patterns such as a synchronising merge can be represented as one composite data connector. An entire workflow can also be represented as one composite workflow component.

Workflow composition: Workflow composition is achieved by connecting output and input of the workflow components through data connectors. These data connectors can be atomic communication channels, composite channels representing control flow patterns such as split or merge or in its most abstract form all interaction of a workflow can be one composite object with links to all workflow components.

3.2 Workfbw Design

Workflow design can be bottom up, top down, automatic or done by hand. The design space for workflow design consists of all possible workflow components. To efficiently find a grounded design which satisfies the problem definition, the design space needs to be constrained. By grounded design is meant, a design which when represented in its most atomic MPMD form only consists of existing implemented workflow components. In bottom up design the problem definition constrains the atomic workflow components which can be used in the first and last step of the workflow due to the fact that the inputs of the first step and outputs of the last step must match those of the problem definition. The outputs of the first step and inputs of the last then form the constraints for the steps which can be used in between. To formally verify whether this workflow satisfies the problem definition and behaves correctly, all atomic workflow components and their connectors have to be taken into account. This is to be expected when no part of this workflow has been constructed before, however when there is the possibility of reusing previous designs formally verifying the reused parts which were already verified previously is inefficient. The workflow design lattice introduced earlier in this section offers a solution. By using the property of compositionality previously created workflows and connectors can be abstracted to single components. Compositionality can be used as the mechanism to abstract both data and process in the workflow design lattice. If these abstracted components have been formally verified, their internal actions will not have to be verified again when they are reused in a new design. They have been shown to satisfy their problem definition, thus this can be used for defining the composite component leaving out all the details of what happens internally.

Multiple execution models: When creating a representation of a workflow with multiple models of execution(illustrated in Figure 6) one can first create the workflow using execution model A and leave the sub problem which can only be solved under execution model B as a problem definition. As an example, let A be a data-flow based system and B a less constraint control flow system which allows a user to steer execution. By allowing this, system B possibly violates the determinacy constraint of system A. Thus when in system A a sub workflow is performed by system B and in this sub workflow the user can steer execution in such a way that for the same input different output actions are possible the constraints of the workflow in A (WFA) are violated. Then the composition of WFA and WFB is not valid. This is clearly an undesirable situation, for which the simple solution is not using system B within

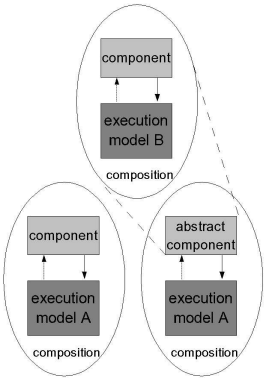

Figure 6. checking whether sub workflow with different execution model implements sub problem description

A. In practise however it can often be the case that while the execution model B allows the user to steer execution in a non-determinate manner, the actual sub workflow under B does not violate determinacy. It can therefore be useful to test for this case.

3.3 Requirements for Formalism

The formalism in which the approach to workflow problems outlined in this section will be performed needs to have certain properties. First of all it needs to be able to abstract both details of process and communication through composition. It also needs to be expressive enough to express all possible workflow patterns, but also all possible processes which can be workflow components. Practical tools such as simulators, model checkers and theorem provers have to be available for such a formalism. Preferably these tools are already tailored towards workflows.

4 Formalisms

In this section we will introduce the formalisms and give a comparison of their features, in the discussion we will address the suitability of the formalisms to the problems introduced in the previous section.

Petri Nets are based on the work in the Phd thesis of Carl Adam Petri in 1962 [21]. His thesis deals with the asynchronous communication between components of a computer system. In its basic form Petri Nets are a graph based formalism, consisting of places transitions input and output functions. This is the starting point for numerous extensions which give Petri Nets additional properties such as Turing completeness [20], explicit data through "coloured" Petri Nets [20], as well as hierarchy [11,28]. Petri Nets have been used frequently as a formalism in workflow research [3].

I/O Automata were first introduced by Lynch and Tuttle in 1989 [14] and have been used for the study of concurrent computing problems. They form a labelled state transition system consisting of a set of states, a set of actions divided into input- internal- and output actions and a set of transitions which consists of triples of state, action and state. In previous work we have argued that I/O Automata are suitable as a workflow formalism [24].

Turing Machines were introduced by Alan Turing in 1936 [25]. They consist of a tape for storing data, a head for reading and writing, a table of instructions and a (finite) state register which stores the state of the table. Turing machines is a very general formalism which we have used in previous work to reason about workflows [23].

Constraint Automata were introduced in 2003 by Arbab, Sirjani, Rutten and Baier [6], as a means of providing formal semantics and analysis of component connectors. Component connectors are a means of connecting software components, this work is very similar to workflow design.Constraint Automata are similar to I/O Automata but with some important differences, for instance Constraint Automata are not input enabled and do not follow a strictly time synchronous approach, these differences will be discussed in more detail later.

π **Calculus** was introduced by Robert Milner in 1989 [17]. It belongs to the family of process calculi, just as λ Calculus and CCS. The specific task it tries to address is describing concurrent processes whose configuration can change during computation. It has been used for describing business workflows [22], there has been some discussion about its suitability particularly versus Petri Nets [26].

4.1 Overview

Turing equivalent

A formalism is Turing equivalent when it is able to represent all possible computational processes, Turing machines and λ Calculus are the classic examples. We were unable to find proof whether I/O Automata are Turing equivalent although this does seem likely given the amount of freedom allowed in defining actions and transitions. Petri Nets are only Turing equivalent when they are expanded to include a zero test arc [20].

graphical representation

For purposes of dissemination it is nice to have a standard graphical representation. Petri Nets have a graphical representations which are commonly used and are well suited to representing workflows. For Turing machines a standard representation does exist that visualises the tape and write head. This is not very well suited as a workflow representation. I/O Automata have no agreed upon visual representation, For both π Calculus and Constraint Au-

	Turing Machine	I/O Automata	Petri Nets	Constraint Automata	π Calculus
Turing equivalent	yes	unknown	only with zero test	yes	yes
graphical representation	not suited to WF	no	yes	yes	yes
data	explicit	implicit	implicit	explicit	explicit
process	explicit	explicit	explicit	explicit	explicit
orientation	process	process	process	data interaction	process
non determinism	in special case	supported	supported	supported	supported
implicit multiple instances	not possible	not possible	possible	not possible	not possible
compositional	supported	supported	supported	supported	supported
simulator	available	available	wf specifi c	wf specifi c	available
model checker	available	available	wf specifi c	wf specifi c	available
theorem prover	yes	yes	no	no	yes
workfbw patterns	not yet shown	not yet shown	many shown	many shown	many shown

Table 1. Features of formal workflow models compared

tomata visual representations suitable for workflows are available

data

In Turing machines data is written explicitly to the tape, in both I/O Automata and Petri Nets data is implicit in the state.Unless for Petri Nets an extension such as coloured Petri Nets [20] are used which explicitly model data. In Constraint automata data is explicitly driving the execution through the use of Timed Data Streams. Within π Calculus data is transferred explicitly over named channels

process

Processes are modelled explicitly in Turing machines through explicit instructions for reading an writing. In both Petri Nets and I/O Automata processes are modelled by explicit transitions, while in π Calculus and Constraint Automata processes are modelled implicitly through the data they communicate.

non determinism

Turing machines are deterministic by definition and can therefore not directly represent non-deterministic processes, a Turing machine can always simulate a non-deterministic process. A Turing equivalent variation, that can represent non-deterministic processes, called non-deterministic Turing machines exists. All other formalisms can directly model non-determinism

implicit multiple instances

The execution of a Petri net comprises multiple instances of the same Petri net, this can occur implicitly through the structure of the Petri net, while in other formalisms this has to be specified explicitly

compositional

Compositionality is a property where in compositions of elements one element can be used to represent another underlying composition consisting of multiple elements. Turing machines support this, as a set of Turing machines can always be simulated on a single Turing machine. Many different ways of performing hierarchical compositions exist as extensions to Petri Nets (for instance [11, 28]). Compositionality is supported by I/O Automata with some restrictions on which elements can be composed together, this in order to avoid ambiguity during execution. Constraint Automata are fully compositional there are no limits on the composition of either processes or communication. In π Calculus processes are compositional, however there is no way to explicitly provide composition for interaction between processes.

refinement

For I/O Automata it is always possible to more precisely define the actions, states and transition relation of a particular automaton without changing the basic topology of a composition. In Petri Nets it is possible to refine data types by using coloured Petri Nets, an extension of Petri Nets. Refining places and transitions in Petri Nets can be done with another extension [11, 28].

simulator

Using a simulator and a formal description of a workflow one can check for reachability, whether a certain state in a workflow component can be reached, safety, there are no undesirable terminations possible, deadlock, test whether the workflow is free of potential deadlock situations, bottlenecks do certain transitions take a lot of time. The simulator for I/O Automata [1] can perform all tests except the last one as it has not got an explicit notion of time. Constraint Automata are a formal model for connectors in an environment called Reo [2, 6]. This environment can potentially be used for simulation but also as an actual workflow engine. For π Calculus there are several simulators developed for Bio Chemical processes [1] they are not suited to workflow modelling. A company called Intalio is developing Business process workflow tools based on π Calculus, which will include a simulator at some point in

[1] http://www.wisdom.weizmann.ac.il/ biopsi/psi.htm

the future. Many different simulators exist both for Petri Nets and Turing machines, including ones for timed Petri Nets which can deal with bottlenecks.

model checker

Model checking tools are available for I/O Automata [1], Petri Nets [27], Constraint Automata [10], and π Calculus [29]. The model checker for I/O Automata does so in a compositional way, where as the Petri net model checker dos not do this as of yet [27].

theorem prover

Theorem provers are currently available for I/O Automata [18]. Several theorem provers for Pi Calculus also exist, such as [16], they are however still work in progress. Work on a theorem prover for use with Petri Nets specifically aimed at business workflows is in progress[2]

workflow patterns

Workflow patterns which demonstrate the ability of a formalism to succinctly express existing patterns in (business)workflows have been demonstrated in great detail at http://www.workflowpatterns.com for Petri Nets. At http://www.pi-workflow.org/ most of these patterns have been reproduced using π Calculus. At http://homepages.cwi.nl/ proenca/webreo/ most patterns are shown for Constraint Automata. Constraint Automata can more easily express some of the more complicated patterns due to the fact that they can represent asynchronous and synchronous interaction simultaneously. For other formalisms these patterns have not yet been shown.

5 Discussion

In this paper we have defined workflow problems which are in need of formal analysis and given an overview of formalisms that can provide this analysis. The question now is which formalism is best suited to these problems. Let us first consider Turing machines, from the overview it is clear that they are expressive enough in theory, they are not a practical solution for modelling real workflows. There are no tools for written for modelling workflows as Turing machines, composition of process nor data is practical when modelling real problems. It is theoretically always possible to merge tapes or simulate multiple Turing machines on one machine, this is not a practical solution when Turing machines accurately model real data or processes. Turing machines have there place in purely theoretical proofs, but not in design approach which results in real workflows. In the overview of formalisms Constraint Automata are oriented to describing interaction between processes, while the other formalisms are oriented towards describing the process. The consequence of this is that the problem definition, the most abstract representation of a workflow, is defined most easily in terms of data interaction for Constraint Automata. For the other formalisms a process description is easier.

We have argued that compositionality can be a very useful property when analysing workflow problems. All formalisms support composition in some form, in Petri Nets it as an extension for which multiple solutions exist, in I/O Automata and Constraint Automata it was included from the outset. The advantage of this is that many of the tools associated with these formalisms also work in a compositional way while for Petri Nets this is not always the case. A Petri net model checker [27] aimed at workflow analysis, for instance does not yet work in a compositional way.

Defining data connectors in a compositional way using a process oriented formalism is more difficult, as data connections need to be defined in terms of processes. This is especially true when data is already explicit which is the case for π Calculus. In Petri Nets a composition of only places, or only processes is not possible. In I/O Automata it is possible to abstract data connectors and processes separately if one defines all data connectors in terms of processes. Even then there are still some limitations on the compositionality of I/O Automata, not all automata are allowed in the same composition to avoid ambiguity during execution. In practise for this could be avoided by being careful with the naming of the actions inside I/O Automata.

For Petri Nets, π Calculus and Constraint Automata the expressiveness has been shown through workflow patterns, for business workflow management systems there are overviews of which system supports what patterns. This is a very significant help when defining the model of computation for a certain workflow. A similar analysis for scientific workflow management systems should be done. These will probably yield a less diverse set of patterns as most are data flow based. For the other formalisms accurately describing the model of computation will be more work. There is some past work which can be of help though. In [13] it is shown how the Kahn principle can be modelled using I/O Automata. The main constraints are that the automata have to be deterministic and all connections are one to one. Kahn process networks are similar to many of the data flow models of computation used in scientific workflow management systems.

As mentioned in section 3.3 it is preferable that any tools associated with a formalism exist in a form suitable to workflows. This is most certainly the case for Petri Nets which have long been used for reasoning about workflows. Reo, the tool associated with Constraint Automata is developed for distributed systems composed of heterogeneous mobile black-box components. While these systems encompass more than just workflows, they do fit well within that description. Tools for Constraint Automata are thus very well

[2] http://wwwis.win.tue.nl/ movebp/description.html

suited to workflows. The tools for I/O Automata are more general still and lack specific workflow features. Workflow specific tools for π Calculus are still largely under development.

6 Conclusions & future work

We presented an approach to workflow design at multiple levels of data and process abstraction. It is desirable to formally reason about workflows in this way and prove the correctness of complicated workflows. For instance those workflows involving multiple workflow engines and execution models. We can determine which workflow systems can safely be used to provide sub workflows for other workflow systems. In case this is not safe in general we can determine of a specific sub workflow can still be used safely. An overview was given of formalisms which can be used for workflow design. From this overview it becomes clear that Constraint Automata seem most suited to our approach at this time, they have the best support for compositionality, they are the most expressive and have tools available suitable for use with workflows. π Calculus has its place for different design approaches where there is less emphasis on compositionality of data interaction. I/O Automata offer almost the same support composition, but are less suited to expressing data interaction. Furthermore they need a lot of work both in proving expressive capability as well as development of workflow specific tools. Petri Nets are well established as a workflow formalism, but they were not intended to be used compositionally from the outset, which shows in diminished compositional ability both in the formalism itself and associated workflow tools. As future work we intend to use the design method outlined in this paper in combination with Constraint Automata to create an automatic workflow composition tool. We also intend to apply this approach to the workflow bus [30] that is being developed at our research group.

Acknowledgements We would like to thank Farhad Arbab for his valuable input.

This work was carried out in the context of the Virtual Laboratory for e-Science project (www.vl-e.nl). Part of this project is supported by a BSIK grant from the Dutch Ministry of Education, Culture and Science (OC&W) and is part of the ICT innovation program of the Ministry of Economic Affairs (EZ).

References

[1] Ioa language and toolset, http://theory.csail.mit.edu/tds/ioa/.

[2] Reo homepage, http://www.cwi.nl/htbin/reo/view/main/webhome.

[3] W. M. P. V. D. Aalst, A. H. M. T. Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distrib. Parallel Databases*, 14(1):5–51, 2003.

[4] M. Alt, S. Gorlatch, A. Hoheisel, and H.-W. Pohl. A grid workflow language using high-level petri nets. In R. Wyrzykowski, J. Dongarra, N. Meyer, and J. Wasniewski, editors, *Parallel Processing and Applied Mathematics*, volume 3911 of *LNCS*, pages 715–722. Springer-Verlag, 2006.

[5] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludäscher, and S. Mock. Kepler: An extensible system for design and execution of scientific workflows. In *SSDBM*, pages 423–424, 2004.

[6] C. Baier, M. Sirjani, F. Arbab, and J. Rutten. Modeling component connectors in reo by contraint automata. *Science of computer programming*, 61:75–113, 2006.

[7] I. T. Foster, J. Vöckler, M. Wilde, and Y. Zhao. Chimera: A virtual data system for representing, querying, and automating data derivation. In *SSDBM '02: Proceedings of the 14th International Conference on Scientific and Statistical Database Management*, pages 37–46, Washington, DC, USA, 2002. IEEE Computer Society.

[8] M. Gulamali, A. McGough, R. Marsh, N. Edwards, T. Lenton, P. Valdes, S. Cox, S. Newhouse, and J. Darlington. Workflow enactment in iceni. In *In UK e-Science All Hands Meeting*, pages 792–799, 2004.

[9] C. Hylands, E. A. Lee, J. Liu, X. Liu, S. Neuendorffer, Y. Xiong, Y. Zhao, and H. Zheng. Overview of the Ptolemy project. Technical report, University of California at Berkeley, July 2003.

[10] S. Kluppelholz and C. Baier. Symbolic model checking for channel-based component connectors. In *FOCLASA 06*, page tba, 2006.

[11] C. Lakos. Object oriented modelling with object petri nets, 1997.

[12] B. Ludascher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger-Frank, M. Jones, E. Lee, J. Tao, and Y. Zhao. Scientific workflow management and the kepler system. *Concurrency and Computation: Practice and Experience, Special Issue on Scientific Workflows*, 18(10), 08 25 2006.

[13] N. A. Lynch and E. W. Stark. A proof of the kahn principle for input/output automata. *Information and Computation*, 82(1):81–92, 1989.

[14] N. A. Lynch and M. R. Tuttle. An Introduction to Input/Output Automata. *CWI Quarterly*, 2(3):219–246, September 1989.

[15] S. Majithia, M. S. Shields, I. J. Taylor, and I. Wang. Triana: A Graphical Web Service Composition and Execution Toolkit. In *Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, pages 514–524. IEEE Computer Society, 2004.

[16] T. F. Melham. A mechanized theory of the pi-calculus in HOL. *Nordic Journal of Computing*, 1(1):50–76, 1994.

[17] R. Milner. *Communicating and Mobile Systems: the Pi-Calculus*. Springer Verlag, 1991.

[18] T. Ne Win. Theorem-proving distributed algorithms with dynamic analysis. Master's thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA, May 2003.

[19] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li. Taverna: A tool for the composition and enactment of bioinformatics workflows. *Bioinformatics Journal.*, online, June 16, 2004.

[20] J. L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice Hall, Englewood Cliffs, NJ, 1981.

[21] C. A. Petri. Fundamentals of a theory of asynchronous information fbw. In *IFIP Congress*, pages 386–390, 1962.

[22] H. Smith and P. Fingar. Workfbw is just a pi process. *BPTrends, www.bptrends.com*, January 2004.

[23] F. P. Terpstra and P. Adriaans. Designing worklfow components for e-science. In *E-science 2006, 2nd IEEE International Conference on e-Science and Grid Computing, Amsterdam Netherlands*, 2006.

[24] F. P. Terpstra, Z. Zhao, and W. M. P. Adriaans. Towards a formal foundation for aggregating scientifi c workfbws. In *proceedings of ICCS 2007 conference*, page to appear, 2007.

[25] A. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42(2):Eprint. Reprinted in The Undecidable pp.115–154, 1936.

[26] W. van der Aalst. Pi calculus versus petri nets. *BPTrends, www.bptrends.com*, May 2005.

[27] K. van Hee, O. Oanea, N. Sidorova, and M. Voorhoeve. Verifying generalized soundness for workfbw nets. In *Proc. of the 6th International Conference on Perspectives of System Informatics, PSI'2006, Novosibirsk*, pages 231–244, June 2006.

[28] J. Wang, Y. Deng, and M. Zhou. Compositional time petri nets and reduction rules. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 30(4):562–572, 2000.

[29] P. Yang, C. R. Ramakrishnan, and S. A. Smolka. A logical encoding of the π-calculus: model checking mobile processes using tabled resolution. *Int. J. Softw. Tools Technol. Transf.*, 6(1):38–66, 2004.

[30] Z. Zhao, S. Booms, A. Belloum, C. de Laat, and B. Hertzberger. Vle-wfbus: a scientifi c workfbw bus for multi e-science domains. In *E-science 2006, 2nd IEEE International Conference on e-Science and Grid Computing, Amsterdam Netherlands*, 2006.