Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 201 (2008) 3–30

www.elsevier.com/locate/entcs

On the Refinement of Atomic Actions

Richard Banach[1]

School of Computer Science, University of Manchester, Oxford Road, Manchester, M13 9PL, U.K.

Gerhard Schellhorn[2]

Lehrstuhl für Softwaretechnik und Programmiersprachen, Universität Augsburg, D-86135 Augsburg, Germany

Abstract

Inspired by the properties of the refinement development of the Mondex Electronic Purse, we view an atomic action as a family of transitions with a common before-state, and different after-states corresponding to different possible outcomes when the action is attempted. We view a protocol for an atomic action as a computation tree, each branch of which achieves in several steps, one of the outcomes of the atomic action. We show that in this picture, the protocol can be viewed as a relational refinement of the atomic action in a number of ways. Firstly, it yields a 'big diagram' simulation à la ASM. Secondly, it yields a 'small diagram' simulation, in which the atomic action is synchronised with an individual step along each path through the protocol, and all the other steps of the path simulate skip. We show that provided each path through the protocol contains one step synchronised with the atomic action, the choice of synchronisation point can be made freely. We describe the relationship between such synchronisations and forward and backward simulations. We relate this theory to serialisations of system runs containing multiple transactions, and show how existing Mondex refinements embody the ideas developed.

Keywords: Atomic Actions, Protocols, Synchronisation, Forward and Backward Simulation, Refinement, Mondex.

1 Introduction

The Mondex Electronic Purse was developed formally in the mid-1990s using Z refinement. It was one of the first developments to achieve an ITSEC E6

[1] Email: banach@cs.man.ac.uk

[2] Email: schellhorn@informatik.uni-augsburg.de

doi:10.1016/j.entcs.2008.02.013

security rating [7].[3] Rather unusually for a commercial product, a sanitised version of the core of the formal development was made publicly available [26]. Since then it has been a fertile ground for formal methods researchers — the original, human-built proofs of the security properties have been subjected to re-examination by contemporary techniques, and have stood up extremely well to the fiercest tool-based scrutiny achievable today, the first such mechanical verification being [24].

The Mondex formal development featured a refinement proof from an atomic abstract model to a multi-step protocol at the concrete level. The principal component of this refinement proof was a backward simulation from abstract to concrete. At the time of the original development, the development team did try to construct a forward simulation, but were not successful. For a long time it was believed that a forward simulation refinement was impossible. It is by now known that a forward simulation is entirely possible, and more than one has been constructed [3,22,12].

In this paper we explore the wider question regarding possible kinds of simulation for the refinement of an atomic action into a multi-step protocol, in order to settle the matter in the general case. We do this in the simplest possible relational framework in order to avoid complications that would distract from the main point.

In Mondex, the original refinement was done in a $(1, 1)$ manner, i.e. single concrete steps were made to refine single abstract ones. Consequently, since overall, there are more concrete steps than abstract ones, many concrete steps had to refine skip. Of course, one advantage of the $(1, 1)$ strategy is that, in the face of malevolent users or an unpredictable environment, the concrete protocol can be proved to refine the abstract atomic action, no matter how such a user might interrupt the intended playing out of the protocol — since every possible sequence of concrete steps that can be executed, corresponds to some abstract execution, even if it is one consisting entirely of skips.

In this, the original framework, the backward simulation correlated with an early synchronisation, i.e. the single non-trivial abstract step was $(1, 1)$ matched with a step that occurred early in protocol runs. By contrast, the more recently discovered forward simulations correlate with a late synchronisation, namely, the various possible non-trivial abstract steps are $(1, 1)$ matched with steps that occur late in protocol runs.

Given the past uncertainty regarding forward and backward simulations in such contexts, our aim in this paper is to give a general treatment. In Section

[3] Nowadays, national standards like ITSEC have been superseded by the ISO Common Criteria standard [13]. The highest ITSEC level, E6, corresponds to the highest Common Criteria level, EAL7.

2 we outline the operation of our motivating example, the Mondex Purse. In Section 3 we develop a theory of the refinement of a non-deterministic atomic action to a multi-step protocol. This explores the way that the single atomic action can be synchronised with an individual step of the protocol in a $(1, 1)$ refinement, and we see that there are a large number of possibilities for this which we call synchronisation assignments (SAs). We see that SAs are related to the possible choices of forward or backward simulations, according to the manner in which abstract outcomes are related to the details of the SA. In Section 4 we relate the preceding theory of an isolated protocol run to the more global picture needed to embed protocol runs into system runs, and we explore serialisability and the 2-phase property. In the following Section 5 we apply the theory developed to the various refinements of Mondex available today. The final section concludes, and outlines extensions of the work needed to deal with not only some of the more obscure possibilities that Mondex allows, but also more general scenarios where the 'protocol' mental picture is appealing.

2 Mondex: A Motivating Example

Fundamentally, Mondex is a smartcard purse. Since it is a purse, it contains real money, and since it is a smartcard, it contains the money in digital form. This money is designed to be transferable from purse to purse. As for real money, the intention is that such transfers are normally performed in exchange for some desired purpose such as the purchase of goods or services, but equally —just as for real money— it is not the responsibility of the money itself to ensure that the transfer in which it engages is of a genuine nature. The only concern of money in general and of Mondex money in particular, is that it should be unforgeable.

The major objective of the original Mondex development was to develop a protocol for money transfer that ensured that:

(i) Mondex money was unforgeable, even in the face of incomplete execution of the protocol or of malicious behaviour of the environment;

(ii) any full or partial run of the protocol is equivalent to either a successful money transfer, or a traceably (and thus recoverably) lost-in-transit money transfer, or a null action.

These two properties are what make Mondex credible in the face of customer requirements: the first property, unforgeability, gives confidence in the value of Mondex money; while the second property, atomicity, gives comprehensibility when compared with the behaviour of conventional financial transactions. Fig. 1 shows the atomic abstraction that the Mondex protocol ensures, reflecting the three possibilities given in (ii) above. In Fig. 1 the nodes are states, and the arrows are the different atomic actions that the concrete protocol refines.

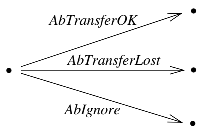

Fig. 1. The Mondex atomic actions.

The essence of the Mondex concrete protocol is illustrated in Fig. 2 in activity diagram style. The source purse is the From purse while the destination purse is the To purse. The protocol begins with the two Start events (initiated from the environment as a result of the purses' owners typing in appropriate instructions at the interface device (the wallet) into which the two purses have been inserted). These are the StartTo event, performed by the To purse, and StartFrom event, performed by the From purse, both of which take their respective purse from the idle state to a 'busy' state: the epr state (expecting payment request) for the From purse, and the epv state (expecting payment value) for the To purse. The StartTo event sends a req message to the From purse. Upon arrival of the req message, the From purse performs a Req event and dispatches the money in a val message to the To purse, itself passing into the epa (expecting payment acknowledgement) state. Upon arrival of the val message, the To purse performs a Val event and sends an ack message to the From purse, itself passing back into the idle state. Receipt of the ack message in the Ack event by the From purse completes the protocol, and the From purse too passes back into the idle state. Note that in Fig. 2, the nodes are now events, edges are states, and arrows are messages.

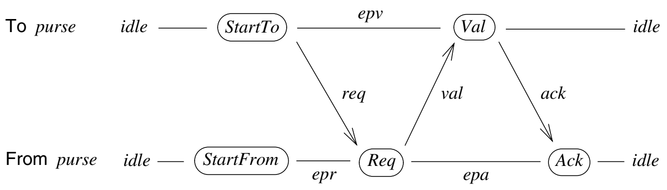

Fig. 2. The Mondex concrete protocol.

The preceding described the workings of a successful run of the protocol. Beyond that, all events after the Start events can be replaced by Abort events, corresponding to runs of the protocol that are unsuccessful for whatever reason. The fact that despite Abort events, the protocol still enjoys the unforgeability and atomicity properties, is what makes Mondex non-trivial theoretically. However, the details of how this comes about do not concern us in this paper.

A further issue is that the Mondex protocol is isolated, i.e. once the protocol has commenced, the two purses take note only of the arrival of the next message expected in the playout of the protocol, and of calls to Abort, ignoring all other messages or calls from the environment and reserving the option of responding to such unexpected events by performing a self-initiated Abort whenever appropriate.

In this paper, rather than being concerned with proving that the atomicity and isolatedness properties are enjoyed by the protocol, we take properties such as these for granted, and instead, take an interest in simulation-theoretic properties —in a general sense, and for their own sake— of the refinement of an atomic action to a protocol with characteristics such as Mondex's. The isolated property makes these simulation-theoretic properties particularly convenient to study.

3 Isolated Atomic Actions and their Protocols

For both protocols and atomic actions, we will specify the transitions involved using a relational approach. The following statements summarise the assumptions we make about this setup.

Assumptions 3.1

(i) Relations are represented by predicates whose variables take values in suitable types.

(ii) Each relation used is deterministic, i.e. for each collection of values for the domain variables of the predicate representing the relation, there is a unique collection of values for the codomain variables that makes the relation true.

(iii) For each relation, for all values of domain and codomain variables that make the relation true, the domain values are reachable from an initial state.

(iv) Where nondeterminism (whether at the atomic or the protocol level) is needed, it is handled by having different relations for different outcomes. We assume nondeterminism is always finite.

(v) Both protocols and atomic actions are represented by computation trees, in which each edge of the computation tree graph corresponds to a (predicate representing a) unique relation. All computation trees are assumed finite.

(vi) For both protocol and atomic action computation trees, the predicatelabelled trees are interpreted within structures which are themselves forests. A choice of initial state for the first step of an atomic or protocol computation tree, picks out a unique tree of the interpreting forest, called the valid subtree.

Thus an atomic action will be specified by a finite collection of deterministic predicates $At_{\mathsf{k}}(u,i,o,u')\quad k=1\ldots$, in which u and u' are (variables denoting) the before- and after- states of the atomic action, i and o are the input and output of the action (these may in fact denote sequences (or more complex structures) of input and output values corresponding to the finer grained events in the protocol, if convenient), and the index k distinguishes the different deterministic outcomes for the same starting conditions. All together, the complete atomic specification of the protocol becomes:

$$Atomic(u,i,o,u') \equiv At_1(u,i,o,u') \vee At_2(u,i,o,u') \vee \ldots \qquad \text{where} \tag{1}$$

$$(\forall u,i \bullet At_{\mathsf{k}}(u,i,o_1,u'_1) \wedge At_{\mathsf{k}}(u,i,o_2,u'_2) \Rightarrow o_1=o_2 \wedge u'_1=u'_2) \tag{2}$$

At the protocol level, the individual steps are described by a collection of deterministic predicates $St_{\mathsf{p}}(v,j,p,v')$ where v and v' are the before- and after-states of the step, j and p are the input and output of the step, and ρ is an identifier which uniquely identifies an edge in the graph of the protocol computation.

N. B. While the decision to represent atomic actions via shallow trees is a natural one, the decision to represent even multistep protocols via deeper trees has consequences that deserve to be spelled out. Protocols can often arrive at 'essentially the same' state via different paths, obtained eg. via interchanges of causally independent steps somewhere in the interior of the protocol. In our formulation, such 'essentially the same' states have to be regarded as different. So our protocol states can be understood as incorporating the full history of the protocol up to the given point. (Such history information is not only convenient here, but is in any case often needed in reasoning about protocols, since protocol properties frequently depend not only on knowing that the protocol has arrived at a certain point, but that certain other things must have necessarily happened prior to that point. Such facts can be trivially extracted from the full history, so our formulation may be regarded as a multipurpose canonical description, useful for things other than just the concerns of this paper. In the next section, we get the opportunity to project out such aspects of the protocol state as deserve to be regarded as unrealistic.)

Another aspect that should be discussed is I/O. At the atomic level, the I/O for the single step that takes place must inevitably concern the environment, since there is no internal structure to engage in internal communication. At the protocol level however, I/O can either be between the environment and the protocol, or be purely internal to the protocol. In the latter case, the only restriction is that messages must be produced before they can be consumed. There is of course the option of representing messages in flight within a suitable state component —such a state component can model properties of the communication medium, eg. unreliablity— however we do not need to insist on this for the theory here.

(Forward) paths through the computation tree are described by compound predicates:

$$\begin{aligned} &FPath_{\langle\alpha,\beta,\ldots,\gamma\rangle}(v_{\mathsf{I}}, j_1, p_1, v_1, j_2, p_2, v_2, \ldots, v_{\mathsf{t}-1}, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{t}}) \equiv \\ &\quad St_{\alpha}(v_{\mathsf{I}}, j_1, p_1, v_1) \wedge St_{\beta}(v_1, j_2, p_2, v_2) \wedge \ldots \wedge St_{\gamma}(v_{\mathsf{t}-1}, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{t}}) \end{aligned} \qquad (3)$$

in which v_{I} is a possible initial state of the protocol, α labels a possible first step of the protocol, β labels a possible successor step of the α step of the protocol, and so on. As (3) indicates, if a step has a successor, the before-state of the successor must match the after-state of its predecessor. The length of the sequence of labels in the subscript of $FPath_{\langle\alpha,\beta,\ldots,\gamma\rangle}$ must match both the number of inputs and outputs, and be one less than the number of states, in the argument list.

Maximal paths arise in the obvious way:

$$\begin{aligned} &MPath_{\langle\alpha,\beta,\ldots,\gamma\rangle}(\ldots) \equiv FPath_{\langle\alpha,\beta,\ldots,\gamma\rangle}(\ldots) \\ &\quad \wedge (\langle\alpha,\beta,\ldots,\gamma\rangle \text{ has no proper extension in the computation tree}) \end{aligned} \qquad (4)$$

From maximal and non-maximal paths, we can implicitly define a predicate *BPath* (backward paths) that describes extensions of non-maximal forward paths:

$$\begin{aligned} &MPath_{\langle\alpha,\beta,\ldots,\gamma,\delta,\square\ldots,\zeta\rangle}(v_{\mathsf{I}}, j_1, p_1, v_1, \ldots, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{t}}, j_{\mathsf{t}+1}, p_{\mathsf{t}+1}, v_{\mathsf{t}+1} \ldots, v_{\mathsf{F}}) \equiv \\ &\quad FPath_{\langle\alpha,\beta,\ldots,\gamma\rangle}(v_{\mathsf{I}}, j_1, p_1, v_1, \ldots, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{t}}) \\ &\quad \wedge BPath_{\langle\delta,\square\ldots,\zeta\rangle}(v_{\mathsf{t}}, j_{\mathsf{t}+1}, p_{\mathsf{t}+1}, v_{\mathsf{t}+1} \ldots, v_{\mathsf{F}}) \end{aligned} \qquad (5)$$

In (5), v_{F} is a possible final state of the protocol.

Finally, maximal paths give rise to the predicate $Protocol(v_{\mathsf{I}}, js, ps, v_{\mathsf{F}})$, where v_{F} is a possible final state of the protocol,[4] given by taking the disjunction over all maximal paths, existentially quantifying all intermediate states, and repackaging the inputs and outputs into sequences:

[4] Initial and final states of the protocol coincide exactly with the root and leaf states of the protocol computation tree.

$$Protocol(v_{\mathsf{I}}, js, ps, v_{\mathsf{F}}) \equiv$$

$$\bigvee_{\left\{\substack{\text{maximal}\\ \langle\alpha,\beta,\ldots,\gamma\rangle}\right\}} \left(\begin{array}{l} (\exists\ j_1, p_1, v_1, j_2, p_2, v_2, \ldots, v_{\mathsf{t}-1}, j_{\mathsf{t}}, p_{\mathsf{t}} \bullet \\ \quad MPath_{\langle\alpha,\beta,\ldots,\gamma\rangle}(v_{\mathsf{I}}, j_1, p_1, v_1, j_2, p_2, v_2, \ldots, v_{\mathsf{t}-1}, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{F}}) \\ \quad \wedge\ js = \langle j_1, j_2, \ldots, j_{\mathsf{t}}\rangle \wedge ps = \langle p_1, p_2, \ldots, p_{\mathsf{t}}\rangle) \end{array} \right)$$

The fact that the protocol implements the atomic action is captured by having a retrieve relation $R(u, v)$ (which is required to be a function from protocol states v to atomic states u), and input and output relations $Input(i, js)$ and $Output(o, ps)$, such that the following ASM-style [6] 'big-step' proof obligation holds:

$$\begin{array}{l} Protocol(v_{\mathsf{I}}, js, ps, v_{\mathsf{F}}) \Rightarrow \\ (\exists\ u_{\mathsf{I}}, i, o, u_{\mathsf{F}} \bullet \\ R(u_{\mathsf{I}}, v_{\mathsf{I}}) \wedge Input(i, js) \wedge Atomic(u_{\mathsf{I}}, i, o, u_{\mathsf{F}}) \wedge Output(o, ps) \wedge R(u_{\mathsf{F}}, v_{\mathsf{F}})) \end{array}$$

We further require that the 'big-step' retrieve relation $R(u, v)$ is 'not too big,' i.e. it concerns just the 'states of interest' for the overall protocol, i.e. the initial and terminal states:

$$R(u, v) \Rightarrow (\exists\ js, ps, \tilde{v}\ \bullet Protocol(v, js, ps, \tilde{v}) \vee Protocol(\tilde{v}, js, ps, v)) \tag{6}$$

Conditions (4) and (6) ensure that the hypotheses and conclusions of the big-step PO are valid exactly when the simulation predicate Σ:

$$\begin{array}{l} \Sigma(u_{\mathsf{I}}, i, o, u_{\mathsf{F}}, v_{\mathsf{I}}, js, ps, v_{\mathsf{F}}) \equiv \\ \quad Atomic(u_{\mathsf{I}}, i, o, u_{\mathsf{F}}) \wedge Protocol(v_{\mathsf{I}}, js, ps, v_{\mathsf{F}}) \\ \quad \wedge\ R(u_{\mathsf{I}}, v_{\mathsf{I}}) \wedge Input(i, js) \wedge Output(o, ps) \wedge R(u_{\mathsf{F}}, v_{\mathsf{F}}) \end{array} \tag{7}$$

is true in the given types.

Now that we have connected together the atomic and finegrained descriptions of the protocol, our aim is to develop a general way of seeing how some individual step of a maximal path may be viewed as refining the atomic action, and the consequences of such a view. First we develop some technical machinery in the shape of past and future oriented retrieve relations. Then we introduce synchronisation assignments, which delimit exactly how the choices of individual step within the protocol computation tree may be made. Finally we explore the consequences of these choices for proving the refinement via forward and backward simulation.

From these ingredients we get the ‘past oriented’ retrieve relation R^{P}:

$$R^{\mathsf{P}}(u_{\mathsf{I}}, v_{\mathsf{t}}) \equiv (\exists\ v_{\mathsf{I}}, j_1, p_1, v_1, \ldots, j_{\mathsf{t}}, p_{\mathsf{t}}, \langle \alpha, \beta, \ldots, \gamma \rangle \bullet \\ R(u_{\mathsf{I}}, v_{\mathsf{I}}) \wedge FPath_{\langle \alpha,\beta,\ldots,\gamma \rangle}(v_{\mathsf{I}}, j_1, p_1, \ldots, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{t}})) \qquad (8)$$

and the ‘future oriented’ retrieve relation R^{F}:

$$R^{\mathsf{F}}(u_{\mathsf{F}}, v_{\mathsf{t}}) \equiv (\exists\ j_{\mathsf{t}+1}, p_{\mathsf{t}+1}, v_{\mathsf{t}+1} \ldots, v_{\mathsf{F}}, \langle \delta, \epsilon \ldots, \zeta \rangle \bullet \\ BPath_{\langle \delta,\epsilon\ldots,\zeta \rangle}(v_{\mathsf{t}}, j_{\mathsf{t}+1}, p_{\mathsf{t}+1}, v_{\mathsf{t}+1} \ldots, v_{\mathsf{F}}) \wedge R(u_{\mathsf{F}}, v_{\mathsf{F}})) \qquad (9)$$

It is easy to show the following:

Proposition 3.2

$$R^{\mathsf{P}}(u_{\mathsf{I}}, v_{\mathsf{t}}) \wedge R^{\mathsf{F}}(u_{\mathsf{F}}, v_{\mathsf{t}}) \Rightarrow (\exists\ i, o \ \bullet\ Atomic(u_{\mathsf{I}}, i, o, u_{\mathsf{F}})) \qquad (10)$$

$$R^{\mathsf{P}}(u_{\mathsf{I}}, v_{\mathsf{t}}) \Rightarrow (\exists\ i, o, u_{\mathsf{F}} \ \bullet\ Atomic(u_{\mathsf{I}}, i, o, u_{\mathsf{F}}) \wedge R^{\mathsf{F}}(u_{\mathsf{F}}, v_{\mathsf{t}})) \qquad (11)$$

$$R^{\mathsf{F}}(u_{\mathsf{F}}, v_{\mathsf{t}}) \Rightarrow (\exists\ u_{\mathsf{I}}, i, o \ \bullet\ R^{\mathsf{P}}(u_{\mathsf{I}}, v_{\mathsf{t}}) \wedge Atomic(u_{\mathsf{I}}, i, o, u_{\mathsf{F}})) \qquad (12)$$

The proofs are similar to the proofs of the more interesting following result:

Theorem 3.3

$$R^{\mathsf{P}}(u_{\mathsf{I}}, v_{\mathsf{t}-1}) \wedge St_{\mathsf{p}}(v_{\mathsf{t}-1}, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{t}}) \wedge R^{\mathsf{F}}(u_{\mathsf{F}}, v_{\mathsf{t}}) \Rightarrow (\exists\ i, o, js^{\mathsf{P}}, js^{\mathsf{F}}, ps^{\mathsf{P}}, ps^{\mathsf{F}} \\ \bullet\ Input(i, js^{\mathsf{P}}::\langle j_{\mathsf{t}} \rangle::js^{\mathsf{F}}) \wedge Atomic(u_{\mathsf{I}}, i, o, u_{\mathsf{F}}) \\ \wedge Output(o, ps^{\mathsf{P}}::\langle p_{\mathsf{t}} \rangle::ps^{\mathsf{F}})) \qquad (13)$$

$$R^{\mathsf{P}}(u_{\mathsf{I}}, v_{\mathsf{t}-1}) \wedge St_{\mathsf{p}}(v_{\mathsf{t}-1}, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{t}}) \Rightarrow (\exists\ i, o, u_{\mathsf{F}}, js^{\mathsf{P}}, js^{\mathsf{F}}, ps^{\mathsf{P}}, ps^{\mathsf{F}} \\ \bullet\ R^{\mathsf{F}}(u_{\mathsf{F}}, v_{\mathsf{t}}) \wedge Input(i, js^{\mathsf{P}}::\langle j_{\mathsf{t}} \rangle::js^{\mathsf{F}}) \wedge Atomic(u_{\mathsf{I}}, i, o, u_{\mathsf{F}}) \\ \wedge Output(o, ps^{\mathsf{P}}::\langle p_{\mathsf{t}} \rangle::ps^{\mathsf{F}})) \qquad (14)$$

$$St_{\mathsf{p}}(v_{\mathsf{t}-1}, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{t}}) \wedge R^{\mathsf{F}}(u_{\mathsf{F}}, v_{\mathsf{t}}) \Rightarrow (\exists\ u_{\mathsf{I}}, i, o, js^{\mathsf{P}}, js^{\mathsf{F}}, ps^{\mathsf{P}}, ps^{\mathsf{F}} \\ \bullet\ R^{\mathsf{P}}(u_{\mathsf{I}}, v_{\mathsf{t}}) \wedge Input(i, js^{\mathsf{P}}::\langle j_{\mathsf{t}} \rangle::js^{\mathsf{F}}) \wedge Atomic(u_{\mathsf{I}}, i, o, u_{\mathsf{F}}) \\ \wedge Output(o, ps^{\mathsf{P}}::\langle p_{\mathsf{t}} \rangle::ps^{\mathsf{F}})) \qquad (15)$$

Proof. For (13), from $R^{\mathsf{P}}(u_{\mathsf{I}}, v_{\mathsf{t}-1})$ we know that there is a path through the computation tree from an initial v_{I} to $v_{\mathsf{t}-1}$, satisfying (3), and such that $R(u_{\mathsf{I}}, v_{\mathsf{I}})$ holds. Evidently $St_{\mathsf{p}}(v_{\mathsf{t}-1}, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{t}})$ extends that path. From $R^{\mathsf{F}}(u_{\mathsf{F}}, v_{\mathsf{t}})$ we know that there is a completion of this path to a maximal path from v_{I} to some final v_{F}. This maximal path enables us to derive $R(u_{\mathsf{F}}, v_{\mathsf{F}})$, and provides the witnessing $js^{\mathsf{P}}, js^{\mathsf{F}}, ps^{\mathsf{P}}, ps^{\mathsf{F}}$ so that with $j_{\mathsf{t}}, p_{\mathsf{t}}$ we can assemble $js = js^{\mathsf{P}}::\langle j_{\mathsf{t}} \rangle::js^{\mathsf{F}}$ and $ps = ps^{\mathsf{P}}::\langle p_{\mathsf{t}} \rangle::ps^{\mathsf{F}}$, and then assert $Protocol(v_{\mathsf{I}}, js, ps, v_{\mathsf{F}})$.

Since we have $Protocol(v_{\mathsf{I}}, js, ps, v_{\mathsf{F}})$, we can apply (4). The conclusions of (4) yield $R(\tilde{u}, v_{\mathsf{I}})$ for some $\tilde{u}$; and since R is functional, we must have $u_{\mathsf{I}} = \tilde{u}$. The conclusions of (4) also yield $Atomic(u_{\mathsf{I}}, i, o, \tilde{u}')$ and $R(\tilde{u}', v_{\mathsf{F}})$ for some $\tilde{u}'$. Again, since R is functional, we must have $u_{\mathsf{F}} = \tilde{u}'$. From $Protocol(v_{\mathsf{I}}, js, ps, v_{\mathsf{F}})$ we can also deduce $Input(i, js)$ and $Output(o, ps)$.

For (14), the argument is similar except that we do not have to use the functional nature of R to argue $u_{\mathsf{F}} = \tilde{u}'$, since u_{F} is existentially quantified in the conclusion.

For (15), we note first that by Assumptions 3.1.(iii), v_{t} is reachable from some initial v_{I}. We use this to assert a u_{I} such that $R^{\mathsf{P}}(u_{\mathsf{I}}, v_{\mathsf{t}})$ holds, after which we argue as for case (13). We are done. □

Proposition 3.3 is a crucial observation, since it enables an arbitrary protocol step $St_{\mathsf{p}}(v_{\mathsf{t}-1}, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{t}})$ to be singled out and made to correspond with a suitable abstract one $Atomic(u_{\mathsf{I}}, i, o, u_{\mathsf{F}})$. For such a $St_{\mathsf{p}}(v_{\mathsf{t}-1}, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{t}})$ step, let Outcomes$(St_{\mathsf{p}}, u_{\mathsf{I}})$ (with $v_{\mathsf{t}-1}, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{t}}$ understood) be given by:

$$\begin{aligned}&\text{Outcomes}(St_{\mathsf{p}}, u_{\mathsf{I}}) = \\ &\quad \{u_{\mathsf{F}} \mid (\exists\, v_{\mathsf{F}} \bullet R^{\mathsf{P}}(u_{\mathsf{I}}, v_{\mathsf{t}-1}) \wedge St_{\mathsf{p}}(v_{\mathsf{t}-1}, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{t}}) \wedge R^{\mathsf{F}}(u_{\mathsf{F}}, v_{\mathsf{t}}))\}\end{aligned} \tag{16}$$

and OD$(St_{\mathsf{p}}, u_{\mathsf{I}})$ (outcome determinism of St_{p}, given u_{I}) be given by:

$$\text{OD}(St_{\mathsf{p}}, u_{\mathsf{I}}) = |\ \text{Outcomes}(St_{\mathsf{p}}, u_{\mathsf{I}})\ | \tag{17}$$

If OD$(St_{\mathsf{p}}, u_{\mathsf{I}}) = 1$ we say that St_{p} is outcome deterministic at u_{I} (St_{p} is OD at u_{I}), whereas if OD$(St_{\mathsf{p}}, u_{\mathsf{I}}) > 1$ we say that St_{p} is outcome nondeterministic at u_{I} (St_{p} is ON at u_{I}).

Definition 3.4 Given an initial v_{I}, a synchronisation assignment (SA(v_{I})) for the relevant valid subtree of a protocol computation tree is a subset of its steps, such that for each maximal path through the valid subtree from v_{I}, exactly one of its steps is in SA(v_{I}). Steps in SA(v_{I}) are called SA steps.

Fig. 3 shows a synchronisation assignment. The many-level computation tree at the bottom has thickened arrows which are the elements of the SA. The atomic action is at the top and plays no specific part in the SA itself. Dashed arrows show the functional big-step retrieve relation R, while the dotted lines show some pieces from the R^{P} and R^{F} relations, for convenience below.

Definition 3.5 Given a protocol computation tree, an intial state v_{I} for the protocol, the atomic intial state u_{I} such that $R(u_{\mathsf{I}}, v_{\mathsf{I}})$ holds, and a synchronisation assignment for the valid subtree determined by v_{I}, the steps of the valid subtree are classified as follows:

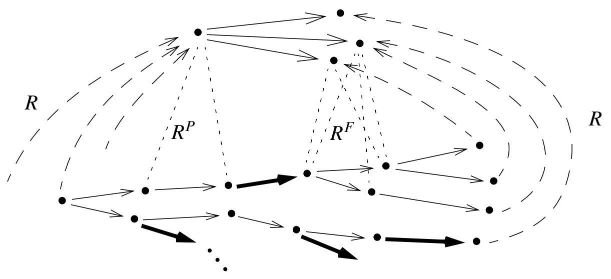

Fig. 3. A synchronisation assignment for a computation tree. The elements of the synchronisation assignment are shown bold.

(i) If a step is in the SA and is OD at u_{I}, it is said to be an outcome deterministic forward synchronisation (ODFS) step.

(ii) If a step is in the SA and is ON at u_{I}, it is said to be an outcome nondeterministic forward synchronisation (ONFS) step.

(iii) If a step is an immediate or later successor of an ONFS step, it is a backward skip (BS) step.

(iv) Every step not covered by (i)-(iii) is a forward skip (FS) step.

This definition shows that every path through the protocol computation tree can be described by the following regular expression:

$$FS* \ ; \ (\ ODFS \ ; \ FS* \ + ONFS \ ; \ BS* \ ) \tag{18}$$

Our aim is to show that when given a big-diagram refinement of an atomic action to a protocol of the kind we have described, if we wish to break the big-diagram refinement down into a collection of small-diagram refinements of zero or one atomic action steps to individual steps of the protocol, one can always use forward simulation reasoning, except for the BS steps. In fact one can use forward simulation reasoning for all steps except branching BS steps (a term explained below), though it comes at a price. Likewise, we have the option of using backward simulation reasoning for all steps if we so wish. We discuss these points later.

Definition 3.6 Assume given an abstract operation $AOp(u,i,o,u')$, a concrete $COp(v,j,p,v')$, and retrieve, input and output relations, $R^1(u,v)$, $In^1(i,j)$ and $Out^1(o,p)$. Then AOp forward simulates COp iff:

$$\begin{aligned} &R^1(u,v) \wedge COp(v,j,p,v') \\ &\quad \Rightarrow (\exists\, i,o,u' \bullet In^1(i,j) \wedge AOp(u,i,o,u') \wedge Out^1(o,p) \wedge R^1(u',v')) \end{aligned} \tag{19}$$

And AOp backward simulates COp iff:

$$\begin{aligned} & COp(v,j,p,v') \wedge R^1(u',v') \\ & \quad \Rightarrow (\exists\, u,i,o \;\bullet\; R^1(u,v) \wedge In^1(i,j) \wedge AOp(u,i,o,u') \wedge Out^1(o,p)) \end{aligned} \tag{20}$$

In both cases, $In^1(i,j)$ and/or $Out^1(o,p)$ can be omitted where there is no input and/or output from AOp and/or COp, as applicable.

Theorem 3.7 Let there be a big-step refinement of an atomic action $Atomic$ to a protocol $Protocol$, given by a retrieve relation R and input and output relations $Input$ and $Output$, so that (4) holds. Let v_{I} be a fixed initial state such that $R(u_{\mathsf{I}}, v_{\mathsf{I}})$ holds, and let $\mathsf{SA}(v_{\mathsf{I}})$ be a synchronisation assignment for the valid subtree rooted at v_{I}. Then the refinement of $Atomic$ to $Protocol$ can be decomposed into single step simulations such that:

(i) If an FS step occurs before an SA step, it is forward simulated by the identity operation on u_{I}.

(ii) If an FS step occurs after an SA step, it is forward simulated by the identity operation on u_{F}, where u_{F} is some outcome of $Atomic$.

(iii) If St_{p} is an SA step, it is forward simulated by $Atomic(u_{\mathsf{I}}, i, o, u_{\mathsf{F}})$ for every u_{F} in $\mathsf{Outcomes}(St_{\mathsf{p}}, u_{\mathsf{I}})$.

(iv) Every BS step is backward simulated by the identity operation on some u_{F}.

Proof. We start by defining R^1, which is:

$$\begin{aligned} R^1(u,v) \equiv (&\exists \text{ a maximal path from some initial } \tilde{v}_{\mathsf{I}}, \text{ and} \\ &((v \text{ precedes an SA step along this path, and } R^{\mathsf{P}}(u,v) \text{ holds}), \vee \\ &(v \text{ follows an SA step along this path, and } R^{\mathsf{F}}(u,v) \text{ holds}))) \end{aligned}$$

Also we must define the single step input and output relations In^1 and Out^1; these however are only needed for the SA steps themselves.

$$\begin{aligned} In^1(i,j) \equiv (&\exists \text{ an SA step } St_{\mathsf{p}}(v_{\mathsf{t}-1}, j, p_{\mathsf{t}}, v_{\mathsf{t}}), js^{\mathsf{B}}, js^{\mathsf{F}} \;\bullet \\ & Input(i, js^{\mathsf{P}} :: \langle j \rangle :: js^{\mathsf{F}})) \end{aligned} \tag{21}$$

$$\begin{aligned} Out^1(o,p) \equiv (&\exists \text{ an SA step } St_{\mathsf{p}}(v_{\mathsf{t}-1}, j_{\mathsf{t}}, p, v_{\mathsf{t}}), ps^{\mathsf{B}}, ps^{\mathsf{F}} \;\bullet \\ & Output(o, ps^{\mathsf{P}} :: \langle p \rangle :: ps^{\mathsf{F}})) \end{aligned} \tag{22}$$

Proving the simulation claims in (i)-(iv) is now rather simple. For (i), (ii) and (iv), since either R^{P} or R^{F} (with the same atomic state) holds for both the before and after states of the FS or BS step, and noting that there is no I/O for these steps, the simulation condition (19) or (20) is readily seen to hold.

For (iii), let $St_{\mathsf{p}}(v_{\mathsf{t}-1}, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{t}})$ be an SA step. We know that $R^{\mathsf{P}}(u_{\mathsf{I}}, v_{\mathsf{t}-1})$ holds for $u_{\mathsf{I}}, v_{\mathsf{t}-1}$, hence $R^1(u_{\mathsf{I}}, v_{\mathsf{t}-1})$ is true, giving the hypotheses of (19). So we must show that the conclusions of (19) hold. For any u_{F} in Outcomes$(St_{\mathsf{p}}, u_{\mathsf{I}})$, we know that $Atomic(u_{\mathsf{I}}, i, o, u_{\mathsf{F}})$ holds. Also we know that $R^{\mathsf{F}}(u_{\mathsf{F}}, v_{\mathsf{t}})$ holds, so $R^1(u_{\mathsf{F}}, v_{\mathsf{t}})$ holds. Since $St_{\mathsf{p}}(v_{\mathsf{t}-1}, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{t}})$ occurs on a maximal path from v_{I} to v_{F}, the totality of inputs along the path, both js^{P} before j_{t}, and js^{F} after j_{t}, will witness that $Input(i, js^{\mathsf{P}}::\langle j_{\mathsf{t}}\rangle::js^{\mathsf{F}})$ holds, giving $In^1(i, j_{\mathsf{t}})$ as required. The reasoning for outputs is similar. So we have all the conclusions of (19), thus completing the proof. □

Since at both abstract and protocol levels, the transpose of the step relation is a partial function, backward simulation is always aligned with a decrease of nondeterminism in both abstract and protocol transition functions. Therefore we get the following (cf. [17]).

Corollary 3.8 Under the assumptions of Theorem 3.7, one can always use single step backward simulations throughout.

We also have the following.

Corollary 3.9 Under the assumptions of Theorem 3.7, suppose there are no BS steps (i.e. all SA steps are OD). Then single step forward simulations can be used throughout.

Obviously, choosing the SA as the last step of each maximal path through the protocol satisfies the hypotheses of Corollary 3.9.

Corollary 3.10 Let v_{F} be a final state accessible from v_{I} such that (4) holds for this choice of $v_{\mathsf{I}}, v_{\mathsf{F}}$ (and suitable other quantities). Let $St_{\mathsf{p}}(v_{\mathsf{t}-1}, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{t}})$ be the SA(v_{I}) step along the (unique) path from v_{I} to v_{F}, $MPath(v_{\mathsf{I}}, \ldots, v_{\mathsf{F}})$. Then the simulation of $MPath(v_{\mathsf{I}}, \ldots, v_{\mathsf{F}})$ by $Atomic(u_{\mathsf{I}}, i, o, u_{\mathsf{F}})$ can be decomposed as follows:

(i) If $St_{\mathsf{p}}(v_{\mathsf{t}-1}, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{t}})$ is an ODFS step, the whole of the simulation of $MPath(v_{\mathsf{I}}, \ldots, v_{\mathsf{F}})$ may be established by inductively forward simulating each step of $MPath(v_{\mathsf{I}}, \ldots, v_{\mathsf{F}})$ from v_{I}, such that:
   (a) predecessors of $St_{\mathsf{p}}(v_{\mathsf{t}-1}, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{t}})$ are forward simulated by the identity operation on u_{I},
   (b) $St_{\mathsf{p}}(v_{\mathsf{t}-1}, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{t}})$ is forward simulated by $Atomic(u_{\mathsf{I}}, i, o, u_{\mathsf{F}})$ where u_{F} is the unique element of Outcomes$(St_{\mathsf{p}}, u_{\mathsf{I}})$,
   (c) successors of $St_{\mathsf{p}}(v_{\mathsf{t}-1}, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{t}})$ are forward simulated by the identity operation on u_{F}.

(ii) If $St_{\mathsf{p}}(v_{\mathsf{t}-1}, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{t}})$ is an ONFS step, the simulation of $MPath(v_{\mathsf{I}}, \ldots, v_{\mathsf{F}}))$ may be established by inductively forward simulating the steps of $FPath(v_{\mathsf{I}}, \ldots, v_{\mathsf{t}})$ from v_{I} up to and including $St_{\rho}(v_{\mathsf{t}-1}, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{t}})$, and inductively backward simulating the steps of $BPath(v_{\mathsf{t}}, \ldots, v_{\mathsf{F}})$ from v_{F} up to v_{t}, such that:

(a) predecessors of $St_{\rho}(v_{\mathsf{t}-1}, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{t}})$ are forward simulated by the identity operation on u_{I},

(b) $St_{\rho}(v_{\mathsf{t}-1}, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{t}})$ is forward simulated by $Atomic(u_{\mathsf{I}}, i, o, u_{\mathsf{F}})$, for each u_{F} in Outcomes$(St_{\rho}, u_{\mathsf{I}})$, establishing $R^{\mathsf{F}}(u_{\mathsf{F}}, v_{\mathsf{t}})$,

(c) successors of $St_{\rho}(v_{\mathsf{t}-1}, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{t}})$ are backward simulated from v_{F} by the identity operation on u_{F}, establishing $R^{\mathsf{F}}(u_{\mathsf{F}}, v_{\mathsf{t}})$.

Why is the above theorem useful? We can give a couple of reasons.

Firstly, it is illuminative. One can be convinced of the correctness of a protocol with respect to an atomic action, without having the details of a refinement already worked out. In such a situation, it may not be clear how to synchronise the atomic action with the lower level description. Theorem 3.7 shows that one can choose this synchronisation relatively freely, within the parameters of allowable synchronisation assignments.

Secondly, once having chosen a synchronisation, it is much easier to write down the 'big-step' retrieve relation and associated input and output relations, than to discover the more finegrained single step ones. Theorem 3.7 shows that with the big-step retrieve relation fixed, the single step ones, R^{P} and R^{F} may simply be calculated. Their generic form needs to be instantiated with the details of the protocol and big-step retrieve relation, and then one must eliminate as many existential quantifiers as possible in order to arrive at a closed form. Making clear that there is such a strategy to follow is a considerable improvement over the hit-and-miss approach one would otherwise need, especially when combined with uncertainty regarding synchronisation.

The theorem also provokes the following considerations.

One can replace some backward simulation by forward simulation. Given a synchronisation assignment, a branching BS step is a BS step $St_{\theta}(v_{\mathsf{s}}, \ldots, v'_{\mathsf{s},1})$ for which there is another BS step $St_{\varphi}(v_{\mathsf{s}}, \ldots, v'_{\mathsf{s},2})$ (with $v'_{\mathsf{s},1} \neq v'_{\mathsf{s},2}$) such that the abstract outcomes $u_{\mathsf{F},1}, u_{\mathsf{F},2}$ corresponding to the completions of the paths from $v'_{\mathsf{s},1}$ and $v'_{\mathsf{s},2}$ are different, $u_{\mathsf{F},1} \neq u_{\mathsf{F},2}$.[5] In such a case, one cannot make a forward simulation inference succeed.

To see this, suppose the first hypothesis of (19) is made true by $R^1(u_{\mathsf{F},1}, v_{\mathsf{s}})$, and the second hypothesis is made true by $St_{\varphi}(v_{\mathsf{s}}, \ldots, v'_{\mathsf{s},2})$. Then the first hypothesis demands that u_{F} be chosen to be $u_{\mathsf{F},1}$, while the second hypothesis demands that u_{F} be chosen to be $u_{\mathsf{F},2}$, a contradiction. This is the standard

5 Since we speak of a BS step, there must be such $u_{F,1} \neq u_{F,2}$, as the nondeterminism in $Atomic(u_I, i, o, u_F)$ has been resolved earlier than at this BS step.

backward simulation counterexample.

In Fig. 3, the SA element along the upper thread of the computation tree is an ONFS step, since it can reach two concrete final states that retrieve to two different abstract outcomes. Accordingly, the two BS steps immediately following it (and the two following the topmost of them along the upper thread) are branching BS steps, since they too can individually reach different concrete final states that retrieve to the two different abstract outcomes. With the dotted lines depicting R^{F}, it is easy to see that these steps illustrate what we have just discussed.

However, if a BS step is **not** branching, i.e. there is only one protocol successor state v'_{S} to v_{S}, then the preceding problem cannot arise since the unique successor cannot force a distinction between the choices for u_{F}. So for nonbranching BS steps, a forward simulation inference will succeed. However, it comes at a price. If a forward simulating BS step immediately follows a backward simulating BS step, the $R^1(u_{\mathsf{F}}, v)$ value at the v state that they share, occurs as a hypothesis in both the backward PO (20) and the forward PO (19). It thus remains as an unproved assumption in the overall single-step verification of the big-step refinement. As such it allows the verification to succeed vacuously. For this reason we phrased Theorem 3.7.(ii) as two inductive processes that meet in the middle, since it is much better to verify some $R^1(u_{\mathsf{F}}, v)$ twice independently, than to leave some other $R^1(u_{\mathsf{F}}, v)$ unproved, thus undermining the whole verification.

Lastly, Theorem 3.7 offers a different strategy for addressing global correctness (see the next section). Normally, to prove a protocol such as the one we have been considering globally correct, one chooses either forward or backward simulation, establishes that each protocol step refines some atomic option or skip, and this then extends to an inductive proof for global executions as a whole. With Theorem 3.7, we can envisage a different approach. We first study the 'big-step' refinement of atomic action to protocol, determining the protocol computation tree and the big-step retrieve relation. Next we choose a suitable synchronisation assignment. Next we determine which combination of forward and backward simulations are appropriate for the synchronisation assignment. Next we calculate the necessary single step retrieve relation, breaking down the big-step refinement into single step refinements. Finally, we determine how runs of the protocol can interleave to make global executions. This alternative approach separates concerns, and in cases where a complex protocol is concerned, may offer some advantages. In any event, the mere awareness of the possibility of such an approach may make the more monlithic standard approach more tractable, since it can show that certain subgoals of the standard approach are achievable in advance. It is to such matters that we now turn.

4 Interleaving Individual Protocol Runs

Thus far, although using language such as 'protocol,' in reality we have only discussed some properties of computation trees. In genuine protocols, various agents interact by performing events and sending/receiving messages etc. We must connect our theory to this world.

The basic idea is that the previous section should be understood as describing (the various possibilities for) a single isolated protocol run, performed by however many agents would be appropriate in practice, with the protocol state recording the full history of the protocol so far (regardless of whether such knowledge is obtainable in principle by the individual agents), and ignoring the rest of the universe. The latter not only regarding other agents/activities in the rest of the universe, but also regarding what the agents of the single protocol run might do both before and after the run itself. So the previous section described an idealised pattern or template for what collections of agents might do over some period of time towards the achievement of some goal described in principle by the atomic action that the protocol implements.

Patterns or templates are normally made to correspond with what happens in the real world by some process of matching, and that is the basis of our approach too. Since we have remarked that our protocol states can include unrealistically detailed history information, our matching process must include a projection mechanism to allow the unrealistic parts to be forgotten. In such a scenario, protocol states that were previously distinct can be matched to the same system state, destroying the previously assumed tree property of the valid subtrees that interpret the protocol. But this is OK. At the system level, we do not need the backward reachability properties that trees guarantee.

Definition 4.1 A system consists of a number of agents, $A_{\mathsf{a}}, A_{\mathsf{b}}, \ldots$ each with its agent state subspace $W_{\mathsf{a}}, W_{\mathsf{b}}, \ldots$. The system state space is $W = W_{\mathsf{a}} \times W_{\mathsf{b}} \times \ldots$. So agent A_{a}'s instantaneous state is some $w_{\mathsf{a}} \in W_{\mathsf{a}}$, and the system's instantaneous state is $w \equiv (w_{\mathsf{a}}, w_{\mathsf{b}}, \ldots)$.

Each agent is a transition system, i.e. the agent can move between different elements of its state space in discrete steps, leaving the state of every other agent unaffected. The enabledness of any agent's transitions is independent of the state of any other agent. Each step can also consume input and produce output, and the I/O policy described in the previous section applies again: i.e. I/O may either be with the environment, or it may be internal to the system and any internal message that is consumed must earlier have been produced.

The transitions are described by a predicate Sy_{A} similar to St in the previous section, where the subscript 'A' refers to the agent performing the step. The transitions of the system as a whole are the interleaved agent transitions of the system's agents.

Definition 4.2 Let S be a system with agents $A_{\mathsf{a}}, A_{\mathsf{b}}, \ldots$. The sequence $\mathcal{T} \equiv \langle w_{\mathsf{I}}, (k_1, A_1, q_1), w_1, (k_2, A_2, q_2), w_2, \ldots \rangle$ is a run of the system iff:

(i) w_{I} is an initial state of the system,

(ii) A_1 is the agent that performs the first step,

(iii) k_1 is the input consumed by A_1 during the first step,

(iv) q_1 is the output produced by A_1 during the first step,

(v) w_1 is the result state of the first step,

(vi) the change of state $w_{\mathsf{I}} \rightarrow w_1$ involves change to the state space W_1 of A_1 only; the state spaces of agents other than A_1 remain unchanged,

(vii) ... and analogously for subsequent system transitions.

Definition 4.3 Let *Protocol* be a protocol in the sense of the previous section. An agent decomposition for the protocol is a decomposition of the protocol state space V into a cartesian product of agent subspaces $V = V_1 \times V_2 \times \ldots$, such that each step of the protocol modifies at most one component in the product, leaving the other components unaltered.

The decomposition into agent subspaces just described, represents the fact that an instantiation of a protocol is normally executed by a number of agents inside a real system. However a real agent in a real system can play many roles during the running of the system, including acting out different roles in different instances of the same protocol at different times. So we need to distinguish the various agent roles in a protocol definition from the different instantiations of these during system runs. The next definition introduces the technical machinery for this.

Definition 4.4 Let $\mathit{Atomic}, \mathit{Protocol}, \ldots$ (with all the attendant machinery) be a protocol implementing an atomic action in the sense of the previous section. We say that system run $\mathcal{T}$ instantiates *Protocol* iff there is a maximal path through the protocol $\mathit{MPath}_{\langle \alpha,\beta,\ldots,\gamma \rangle}(v_{\mathsf{I}}, j_1, p_1, v_1, j_2, p_2, v_2, \ldots, v_{\mathsf{F}-1}, j_{\mathsf{F}}, p_{\mathsf{F}}, v_{\mathsf{F}})$ and there are two maps: τ_{A} and τ_{S} such that:

(i) the signature of τ_{A} is $\tau_{\mathsf{A}} : V \rightarrow W$, and τ_{A} decomposes into a cartesian product of disjoint maps $\tau_{\mathsf{A},\mathsf{l}} : V_{\mathsf{l}} \rightarrow W_{\mathsf{a}_l}$ from each of the agent components of V to distinct agent subspaces of W,

(ii) τ_{S} is an injective map from the steps of the maximal path $\mathit{MPath}_{\langle \alpha,\beta,\ldots,\gamma \rangle}$

to steps of $\mathcal{T}$,

(iii) τ_S is order preserving, i.e. if St_β precedes St_γ in $MPath_{\langle\alpha,\beta,\ldots,\gamma\rangle}$, then $\tau_\mathsf{S}(St_\beta)$ precedes $\tau_\mathsf{S}(St_\gamma)$ in $\mathcal{T}$,

(iv) for each protocol step $St_\beta(v_{\mathsf{t}-1}, j_\mathsf{t}, p_\mathsf{t}, v_\mathsf{t})$ in the domain of τ_S, if V_l is the agent component of V modified during the step, then $\tau_{\mathsf{A},\mathsf{l}}(V_\mathsf{l})$ is the agent subspace modified during the step $\tau_\mathsf{S}(St_\beta(v_{\mathsf{t}-1}, j_\mathsf{t}, p_\mathsf{t}, v_\mathsf{t}))$,

(v) for each protocol step $St_\beta(v_{\mathsf{t}-1}, j_\mathsf{t}, p_\mathsf{t}, v_\mathsf{t})$ in the domain of τ_S, if $\tau_\mathsf{S}(St_\beta(v_{\mathsf{t}-1}, j_\mathsf{t}, p_\mathsf{t}, v_\mathsf{t})) = Sy_{\mathsf{A}_l}(w_{\mathsf{s}-1}, k_\mathsf{s}, q_\mathsf{s}, w_\mathsf{s})$, then $\tau_{\mathsf{A},\mathsf{l}}(v_{\mathsf{t}-1}) = w_{\mathsf{s}-1}$, $j_\mathsf{t} = k_\mathsf{s}$, $p_\mathsf{t} = q_\mathsf{s}$, $\tau_{\mathsf{A},\mathsf{l}}(v_\mathsf{t}) = w_\mathsf{s}$.

(vi) if protocol step St_β modifies V_l and protocol step St_γ is the next protocol step along $MPath_{\langle\alpha,\beta,\ldots,\gamma\rangle}$ that modifies V_l, then no step of $\mathcal{T}$ between $\tau_\mathsf{S}(St_\beta)$ and $\tau_\mathsf{S}(St_\gamma)$ modifies $\tau_\mathsf{A}(V_\mathsf{l})$.

When we want to emphasise the details, we say that system run $\mathcal{T}$ instantiates *Protocol* via $\tau \equiv (\tau_\mathsf{A}, \tau_\mathsf{S})$ at step $\tau_\mathsf{S}(St_\alpha(v_\mathsf{I}, j_1, p_1, v_1))$ of $\mathcal{T}$, where $St_\alpha(v_\mathsf{I}, j_1, p_1, v_1)$ is the initial step in $MPath_{\langle\alpha,\beta,\ldots,\gamma\rangle}$.

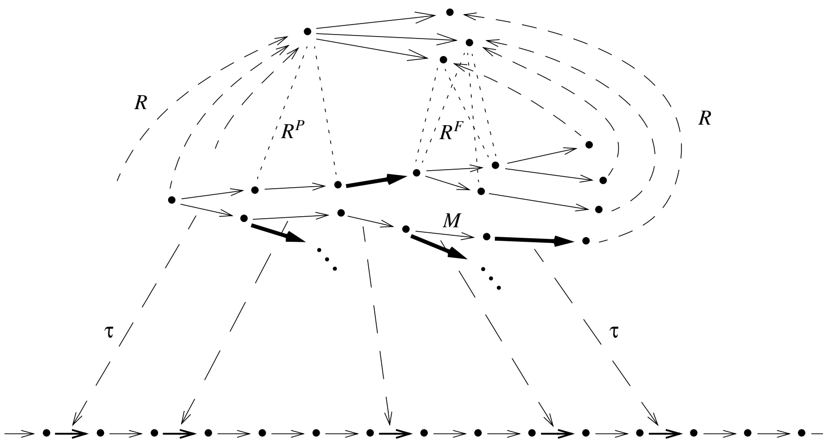

Fig. 4. An atomic action, a protocol which implements it, and a system run containing an instance of a maximal path through the protocol. The steps of the instance are shown bold.

In Fig. 4 we show how a particular maximal path, M say, through the protocol illustrated in Fig. 3, might be mapped, via an instatiation function τ, to a selection of steps in a system run. The system state in the run is now 'real world' state, eschewing the maximal knowledge that the idealised protocol formulation allows. In between the steps of $\tau(M)$, other protocols are being instatiated by other agents, though without interfering with the state of $\tau(M)$, by Definition 4.4.(iv).

Definition 4.5 Let $MPath_{\langle\alpha,\beta,\ldots,\gamma\rangle}$ be a maximal path in *Protocol*. Step

$St_{\mathsf{\beta}}(v_{\mathsf{t}-1}, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{t}})$ of $MPath_{\langle\alpha,\beta,\dots,\gamma\rangle}$ is a first use of agent subspace V_{I} iff: it modifies V_{I}, and no earlier step of $MPath_{\langle\alpha,\beta,\dots,\gamma\rangle}$ modifies V_{I}. Similarly $St_{\mathsf{\beta}}(v_{\mathsf{t}-1}, j_{\mathsf{t}}, p_{\mathsf{t}}, v_{\mathsf{t}})$ is a last use of V_{I} iff: it modifies V_{I}, and no later step of $MPath_{\langle\alpha,\beta,\dots,\gamma\rangle}$ modifies V_{I}. We say that *Protocol* is 2-phase (2P) along $MPath_{\langle\alpha,\beta,\dots,\gamma\rangle}$ iff all first uses of all agent subspaces of *Protocol* precede any last use of any agent subspace of *Protocol* along $MPath_{\langle\alpha,\beta,\dots,\gamma\rangle}$.

Definition 4.6 Let $Sy_{\mathsf{A}}(w_{\mathsf{s}-1}, k_{\mathsf{s}}, q_{\mathsf{s}}, w_{\mathsf{s}})$ and $Sy_{\mathsf{B}}(w_{\mathsf{s}}, k_{\mathsf{s}+1}, q_{\mathsf{s}+1}, w_{\mathsf{s}+1})$ be two successive steps of a run $\mathcal{T}$ of the system. We say that $Sy_{\mathsf{A}}(\dots)$ and $Sy_{\mathsf{B}}(\dots)$ can be commuted iff there is a state $\tilde{w}_{\mathsf{s}}$ such that $Sy_{\mathsf{A}}(\tilde{w}_{\mathsf{s}}, k_{\mathsf{s}}, q_{\mathsf{s}}, w_{\mathsf{s}+1})$ and $Sy_{\mathsf{B}}(w_{\mathsf{s}-1}, k_{\mathsf{s}+1}, q_{\mathsf{s}+1}, \tilde{w}_{\mathsf{s}})$ are valid steps of the system, and the pair $Sy_{\mathsf{A}}(w_{\mathsf{s}-1}, k_{\mathsf{s}}, q_{\mathsf{s}}, w_{\mathsf{s}}); Sy_{\mathsf{B}}(w_{\mathsf{s}}, k_{\mathsf{s}+1}, q_{\mathsf{s}+1}, w_{\mathsf{s}+1})$ can be replaced in $\mathcal{T}$ by $Sy_{\mathsf{B}}(w_{\mathsf{s}-1}, k_{\mathsf{s}+1}, q_{\mathsf{s}+1}, \tilde{w}_{\mathsf{s}}); Sy_{\mathsf{A}}(\tilde{w}_{\mathsf{s}}, k_{\mathsf{s}}, q_{\mathsf{s}}, w_{\mathsf{s}+1})$, yielding $\mathcal{T}'$, where $\mathcal{T}'$ is a valid run.

Lemma 4.7 If $Sy_{\mathsf{A}}(\dots)$ and $Sy_{\mathsf{B}}(\dots)$ as in Definition 4.6, are two successive steps performed by two di�erent agents, then, provided both inputs are available in state $w_{\mathsf{s}-1}$, $Sy_{\mathsf{A}}(\dots)$ and $Sy_{\mathsf{B}}(\dots)$ can be commuted.

Proof. Since $Sy_{\mathsf{A}}(\dots)$ and $Sy_{\mathsf{B}}(\dots)$ are performed by different agents, the two agent subspaces modified by these steps are disjoint, so the changes of state can be swapped, easily yielding the state $\tilde{w}_{\mathsf{s}}$ required by Definition 4.6. If both inputs are available in state $w_{\mathsf{s}-1}$, then the $Sy_{\mathsf{B}}(\dots)$ is enabled in state $w_{\mathsf{s}-1}$ and can be performed first. Since the input to $Sy_{\mathsf{A}}(\dots)$ is not removed by doing $Sy_{\mathsf{B}}(\dots)$, $Sy_{\mathsf{A}}(\dots)$ can follow $Sy_{\mathsf{B}}(\dots)$. That this generates a valid run is now straightforward. □

Since our formulation of a protocol does not consider the protocol's context, the only way that a protocol, as formulated in Section 3, can interact with the rest of the universe is via I/O with the environment. In the system context, this leads to a distinction within the internal system messages, between messages that are produced and consumed by the same protocol instance (which should thus correspond to internal communications of the protocol itself), and those which are produced and consumed by different protocol instances (which should thus correspond to communications with the environment in the protocol model). (System level communications with the environment must of course also correspond with protocol communications with the environment.) Since inter-protocol communications must comply with normal causality considerations, these communications must fit well with the 2-phase property for protocol state components. The next definition introduces the needed technicalities.

Definition 4.8 Suppose given a maximal path $MPath_{\langle\alpha,\beta,\dots,\gamma\rangle}$ of a protocol, which is 2P. An external dependency definition (XDD) for them is, a pair of sets (IDS, ODS) of (not necessarily disjoint) steps. IDS is the input dependency set, and ODS is the output dependency set. A protocol is XDD-normal iff:

(i) all IDS steps occur no later than any ODS step along $MPath_{\langle \alpha,\beta,\ldots,\gamma \rangle}$,

(ii) the producer of every input of every protocol step other than an IDS step is some other step of the same protocol,

(iii) the consumer of every output of every protocol step other than an ODS step is some other step of the same protocol,

(iv) each IDS step occurs no later than any last use of the state,

(v) each ODS step occurs no earlier than any first use of the state.

Definition 4.9 An instantiation of a 2P XDD-normal protocol is called a transaction.

Theorem 4.10 Let $\mathcal{T}_0$ be a run of a system which consists entirely of the steps of transactions of a family of protocols.[6] Then there is a serialisation $\mathcal{T}_\infty$ of $\mathcal{T}_0$, generated by commuting adjacent steps, in which each instantiation occurs as a contiguous series of steps.

Proof. Consider the directed graph Dep_0 whose nodes are the transactions of $\mathcal{T}_0$, and whose edges are given by: $\tau_1 \rightarrow \tau_2$ iff:

(i) an output of an ODS step of τ_1 is an input of an IDS step of τ_2,

(ii) an agent subspace V_l is used by both τ_1 and τ_2, but τ_1's modifications of V_l occur earlier in $\mathcal{T}_0$ than τ_2's.

Claim 4.10.1 Dep_0 is acyclic.

Proof of Claim. Let V be the state space of a transaction τ. Since the last first use of V precedes the first last use of V in τ, and all all IDS steps precede all ODS steps in τ, by Definition 4.8.(iv)-(v), we can deduce that there is a step in τ (which we will call the pivot), that precedes neither the last first use of V nor any IDS step, and simultaneously follows neither the first last use of V nor any ODS step (there are four cases). We identify each transaction in $\mathcal{T}_0$ with (some choice for) its pivot. Since steps are interleaved, there is a total order on the transactions, inherited from that on their pivots.

We show that Dep_0 can be interpreted in the set of pivots, and that each edge in the interpretation is oriented towards the future, yielding the acyclic-

[6] So there is a set of maximal paths through a set of 2P XDD-normal protocols, and a set of instantiations of them in $\mathcal{T}_0$, and the set of steps of $\mathcal{T}_0$ is the disjoint union of these instantiations

ity of Dep_0 immediately. For a Dep_0 edge of type (i), note that it is oriented towards the future by straightforwards causality. So pretending that the requisite message was sent during the producing transaction's pivot step, and pretending that it arrived during the consuming transaction's pivot step can increase its time of flight, but not change its orientation towards the future. For a Dep_0 edge of type (ii), since the pivot steps are contained within the uses of transactions' state, and these are oriented towards the future by (ii), the orientation is preserved in the interpretation. We have our claim. □ □

We serialise $\mathcal{T}_0$ stage by stage. At each stage there are serialised and unserialised transactions. We call the boundary between the serialised and unserialised transactions the horizon. So at the beginning there are no serialised transactions, and the horizon lies just before the first step of $\mathcal{T}_0$. At the n'th stage, which starts with $\mathcal{T}_\mathsf{n}$, whose unserialised transactions comprise Dep_n (a subgraph of Dep_0), we choose an unserialised transaction which is a root of Dep_n, and we serialise it, whereupon its steps —in contiguous sequence— are both appended to the serialised part, and removed from the unserialised part of the partly serialised run, moving the horizon to just beyond the newly serialised steps, and yielding $\mathcal{T}_{\mathsf{n}+1}$ and $Dep_{\mathsf{n}+1}$. If $\mathcal{T}_0$ is infinite, then the serialisation process continues forever, and every finite prefix of $\mathcal{T}_0$ has all its steps eventually included in the serialised part. If $\mathcal{T}_0$ is finite, the process stops when the last transaction of $\mathcal{T}_0$ has been serialised.

Stage n: A root transaction τ_n of Dep_n is chosen. By assumption, all transactions on which τ_n is dependent, whether through the state space, or via τ_n's IDS messages, have been serialised, i.e. their steps lie beyond the horizon. So any step of $\mathcal{T}_\mathsf{n}$ that lies between the horizon and τ_n's first step neither uses any state used by τ_n's first step, nor produces a message consumed by τ_n's first step. So there is no obstacle to commuting the first step of τ_n towards the past until it it arrives immediately after the horizon. Similarly the dependencies for the second step lie either beyond the horizon, or arise from the first step, so the second step of τ_n can be commuted towards the past until it arrives immediately after the first. The process continues until the last step of τ_n has been commuted until it arrives immediately after its predecessor. This yields $\mathcal{T}_{\mathsf{n}+1}$. Transaction τ_n is removed from Dep_n, yielding $Dep_{\mathsf{n}+1}$, and the horizon is moved to just after τ_n's last step. End Stage n. □

The preceding amounts to a sketch of a relatively standard 2-phase serialisation proof process [4,10,5,28]. And once the run has been serialised, it is clear that each transaction of the serialised run is a refinement of its corresponding atomic action via a retrieve function that forgets the part of the system state not relevant to the transaction.

5 Mondex and its Refinements

In this section we reflect on the Mondex protocol, and the extent to which its refinement possibilities correspond to the preceding theory. There are a number of points to be borne in mind.

First of all, our theory has been couched in terms of single transitions (which is less cluttered), whereas Mondex is couched in terms of Z operations [25,8,14]. Thus when we say below that such and such an operation is synchronised with such and such an atomic action, we are refering in bulk to all the transitions of the operation being suitably synchronised with appropriate instatiations of the atomic action.

Secondly, we will restrict our attention to runs of the protocol which commence with the two Start operations, StartFrom and StartTo, in either order, (returning to other possibilities at the end of this section). Refering to Fig. 2, this means that after the two Start operations, the protocol, which is henceforth serial (as is obvious from the causal dependencies of the req, val and ack messages), executes some prefix of the Req-Val-Ack sequence of operations. If it does not complete that sequence, each purse that still has elements of the Req-Val-Ack sequence left to do, performs an Abort operation (replacing the first such unperformed Req-Val-Ack operation left on that purse's agenda), completing the protocol abnormally. Note however that unlike the Req-Val-Ack operations which are causally constrained by the req, val, ack messages, Abort operations are not causally constrained and can occur at any time. Every variation in the order of performing the protocol's operations when Abort events are involved, causes a branching of the computation tree structure, and leads overall, to quite a complex protocol computation tree.

5.1 The Original Mondex Refinement [26]

In [26], the refinement is constructed to synchronise with the atomic description as early as possible, given the assumptions above. Thus the atomic action is synchronised with the Req operation, which refines both AbTransferOK and AbTransferLost. Since the protocol still has plenty of opportunity to fail after the Req operation, the Req operation itself does not fix the outcome, so the refinement, achieved on the basis of a global inductive proof, has to be a backward one. We can visualise to some extent the substructure of Fig. 3 that forces a backward simulation (refered to at the end of Section 3), from Fig. 2, if we add an edge from Req to an Abort, as an alternative to the message towards Val, since the two abstract outcomes are already available at the end of the Req operation. Furthermore, since for a failing transaction the protocol has already angelically chosen to refine AbTransferLost, the Abort operation(s) which actually signal the failure at the protocol level all refine AbIgnore (which is Mondex-speak for an abstract skip).

5.2 The Refinement of Banach et al. [3]

In [3], amongst other things, a synchronisation with the atomic description that occured late was sought, in order to try to get a forward simulation.[7] The natural operation to refine AbTransferOK to is Val, since that is the moment that the money safely arrives at the recipient. However, if the refinement of AbTransferOK is 'obvious,' then the refinement of AbTransferLost is less so. The subtlety lies within the Abort operation. The deeper structure of the Mondex protocol implies that if only one Abort occurs in a transaction, it is harmless, and such an Abort can refine AbIgnore. Only if two Abort operations occur for a transaction, each while its respective purse is in a critical state, has the transaction failed non-trivially, whereupon the transaction needs to refine AbTransferLost. This leads to the decomposition of the Abort operation into cases, depending on the precise role of the operation in the transaction. In the formalism of this paper, the Abort operation of Mondex corresponds to a collection of events which occur at different places in the computation tree of the protocol, and are thus distinguishable.

The case analysis is interesting. The distinction between benign and non-benign instances of Abort is made on the basis of a purse's local state (specifically, on whether the purse is in state epv or epa (non-benign), or in some other state (benign)). However, since two Aborts make one AbTransferLost, we can only refine AbTransferLost to one of the pair — and it has to be the second of the pair, since if only one Abort in a critical state happens, then it turns out to be benign nonetheless. In [3] non-local state information is used to distinguish the first non-benign Abort from the second, and the first is then made to refine AbIgnore while the second refines AbTransferLost.

5.3 The Refinement of Schellhorn et al. [22]

[22] is the second mechanized verification of Mondex using the the KIV theorem prover [19]. While the first [24] used the original backward simulation and data refinement, the second uses abstract state machines (ASMs, [11], [6]) together with ASM refinement and generalized forward simulations ([20]).

The refinement, like [3], synchronizes successful transfers by having Val

[7] Looking forward to some extent to the specific results of this paper —which show that the essentials of a protocol can be understood by discussing the protagonists in isolation— the discussion in [3] was restricted to a world of just two purses, a single From purse and a single To purse.

implement AbTransferOK. But it chooses to synchronize failed transfers at the earliest point possible. This gives two cases for the Req operation, which is the point where the From purse sends money. In the first, the To purse is still ready to receive the money, in which case Req implements AbIgnore. But if the To purse has already aborted then the second case applies, and Req implements AbTransferLost.[8] Instead of having two cases (as in [3]) in which the Abort operation implements AbTransferLost, the design of [22] leaves only one: the case where the To purse aborts in epv after money has been sent.

The different choices for the synchronisation points was one motivation for us to study the general possibilities here. Another one was to provide a general formalization of using past and future simulation relations (R^{P} and R^{F}). Instances of such relations with a schematic encoding into Dynamic Logic are not only used in the case study [22] but also in earlier work. Future simulation relations occur in the correctness proof of ASM refinement [20]. Past simulation relations are used in coupled refinement [9] as noted in [21].

5.4 The Refinements of Haxthausen, George et al. [12]

The two refinements of [12] use the RAISE specification language [27]. They are another mechanized verification of Mondex using the theorem prover PVS [18]. This case study is slightly out of scope of our theory, since it does not start with atomic actions, but with a two step protocol: the first step (called TransferLeft) is a send operation, which nondeterministically chooses between a success and failure, and we call the two cases SendOK and SendFail. After SendOK, there are again two possibilities: receiving may succeed or fail. For symmetry, we call these operations ReceiveOK and ReceiveFail, [12] calls them TransferRight and Abort. Already, the splitting of transactions at the abstract level into send and receive, allows us to keep the balances of abstract and concrete level in perfect synchrony, as is required by RAISE refinement. The two refinements implement TransferLeft with Req and ReceiveOK with Val.

To compare the synchronisation points with our proofs, we have to add an additional refinement of the original abstract Mondex level to the abstract RAISE level. The refinement would have to implement AbTransferOK by the sequence SendOK ;ReceiveOK. AbTransferLost would be implemented by both SendFail and SendOK ;ReceiveFail. Because SendOK is ON, a forward simulation proof would have to synchronize with the last operation of every sequence. Composing the resulting simulation relation with the existing re-

[8] This differs from [3], where the *Abort* of the From purse that is bound to happen in this situation implements *AbTransferLost*.

finements, we find that the synchronization is the one used in [22].

5.5 The refinements of Butler and Yadav

These refinements develop a Mondex-like money transfer protocol using the B4free tool [2]. They will be published as a contribution to [15]. In accordance to the Event-B [1] methodology, the protocol is developed in many small, but easily mechanically provable refinement steps, the simulations being forward simulations. The strategy decomposes the abstract events to facilitate separate refinement of distinct pieces to distinct protocol level operations. Aside from that, it is similar to that of [3] in that failing transfers are refined by Aborts.

Note that with the exception of the original (backward) one, the preceding refinements are all forward simulations when viewed at the individual protocol instance level (cf. Corollary 3.9). As such, and particularly when they are based on $(1, 1)$ refinements, they all readily extend to forward simulation refinements of full system runs — just as the original $(1, 1)$ backward simulation readily extended to a backward simulation refinement for full system runs.

5.6 Other Possibilities

Our general theory shows that even more possibilities than have been discussed above are actually possible. For example, the refinement of [3] could have chosen to refine AbTransferOK to Ack instead of Val, since Val occurs as the last operation of a successful transaction. However, since in general there is a possibility that a transaction succeeds but that the ack message is lost, causing the Ack operation to be replaced by an Abort (which as it turns out is harmless), we infer that in such a refinement there would be a case in which AbTransferOK would have to be refined by Abort!

An alternative to the preceding is to synchronise right at the beginning, with the first (or second) Start event — and there are plenty of hybrid cases, combining aspects from several of the described or suggested refinements arising from the rich structure of the protocol computation tree. We leave the curious reader to work out such scenarios for him- or her- self.

5.7 The Non-2-Phase Fragments

In discussing the preceding refinements, we have always assumed that the two Start operations are performed first. But it could happen that one purse Starts and immediately afterwards Aborts, before the second purse has Started. This spoils the 2P property since the first purse has relinquished its use of its local state before the second purse has claimed its first use. In such a case, either purse may engage in other transactions, changing the local state, after the first purse's Abort and before the second purse's Start.

A remaining possibility is that only one purse Starts, and the other purse merely Aborts, or does nothing. In such a case, even if the other purse's Abort happens after the (inevitable) Abort of the first purse, it is arguable that the protocol is nevertheless 2P, since the other purse's use of its state amounts to no more than skip. Even if one does not accept this argument, it is evident that the breakdown of the 2P property is rather mild.

Dealing formally with such situations requires an extension of our theory. Note though, that even if these situations are not serialisable via the standard 2P technique, the fact that we have $(1, 1)$ refinements of the protocol, guarantees nonetheless that these 'rogue' interleavings preserve atomic semantics.

6 Conclusions and Further Work

In the preceding sections we took the Mondex Electronic Purse —a prime example of a protocol enacted between a number of parties that was designed to achieve the effect of an atomic action— and we looked for a generalisation. We developed a refinement framework based on seeing both the atomic action and protocol as computation trees, and saw that we could choose the way that the atomic action was synchronised with the protocol in a 'small diagram' refinement relatively freely. The properties of the choice, in particular how potential abstract outcomes were related to synchronisation points, was closely related to the prospects for forward and backward simulation at the small diagram level.

We then embedded this formulation of an isolated protocol run in a framework enabling different runs of perhaps different protocols to be interleaved in a natural way. When combined with a fairly standard 2-phase property, these system runs could be serialised, showing that the atomicity abstraction survives.

We then confronted the theory with various refinements for Mondex that have been created in the recent past, and showed that the flexibility regarding synchronisation points was well borne out in these various refinements.

However, although the majority of 'normal' Mondex transactions (including not only successful ones, but also ones that fail in a 'normal' kind of way) are 2-phase —and the modification of the protocol suggested by Schellhorn et al. in [22] in order to design out the possibility of a certain kind of denial of service attack is 2-phase in its entirety— the original Mondex protocol has some (in practice rare, but in theory interesting) non-2-phase parts. A more sophisticated theory is required to handle those situations.

Besides these issues, Mondex is what we called an isolated protocol. That is to say, once the protocol has commenced, the parties engaging in it are fixed, and no intrusion by other agents is contemplated. (In practice, the Mondex purse's local state determines how much notice is taken of which messages from which agents.) Thus it is natural to ask how the theory develops for protocols having state that is genuinely shared between a number of agents, including cases where the number is not necessarily determined at the start of the protocol. Such extensions will also allow the direct modelling of more sophisticated behaviour by the I/O environment than we have contemplated in this paper. (To capture, using the techniques of this paper, I/O behaviour more subtle than the simple delivery of messages injected into the environment, one would have to regard the environment as an agent in its own right, participating in an esential way in protocols.) These directions will be investigated in future work.

References

[1] Abrial, J.-R. and S. Hallerstede, *Refinement, Decomposition, and Instantiation of Discrete Models: Application to Event-B*, Fundamenta Informaticae **21** (2006).

[2] *Clearsy. b4free tool home page.* *www.b4free.com*.

[3] Banach, R., M. Poppleton, C. Jeske and S. Stepney, *Retrenching the Purse: The Balance Enquiry Quandary, and Generalised and (1,1) Forward Refinements*, Fund. Inf. **77** (2007), pp. 29–69.

[4] Bernstein, P. A., V. Hadzilacos and G. N., "Concurrency Control and Recovery in Database Systems," Addison-Wesley, 1987.

[5] Bernstein, P. A. and E. Newcomer, "Transaction Processing," Morgan Kaufmann, 1997.

[6] Börger, E. and R. Stärk, "Abstract State Machines. A Method for High Level System Design and Analysis," Springer, 2003.

[7] Department of Trade and Industry, *Information Technology Security Evaluation Criteria* (1991), http://www.cesg.gov.uk/site/iacs/itsec/media/formal-docs/Itsec.pdf.

[8] Derrick, J. and E. Boiten, "Refinement in Z and Object-Z," FACIT, Springer, 2001.

[9] Derrick, J. and H. Wehrheim, *Using Coupled Simulations in Non-atomic Refinement*, in: D. Bert, J. Bowen, S. King and M. Walden, editors, *ZB 2003: Formal Specification and Development in Z and B* (2003), pp. 127–147.

[10] Gray, J. and A. Reuter, "Transaction Processing," Morgan Kaufmann, 1993.

[11] Gurevich, Y., *Evolving Algebras 1993: Lipari Guide*, in: E. Börger, editor, *Specification and Validation Methods*, Oxford Univ. Press, 1995 pp. 9–36.

[12] Haxthausen, A. E., C. George and M. Schütz, *Specification and Proof of the Mondex Electronic Purse*, in: M. R. C. Xin, Z. Liu, editor, *Proceedings of 1st Asian Working Conference on Verified Software, AWCVS'06, UNU-IIST Reports 348, Macau*, 2006.

[13] International Standards Organisation, *Common criteria for information security evaluation* (2005), ISO 15408, v. 3.0 rev. 2.

[14] ISO/IEC 13568, "Information Technology – Z Formal Specification Notation – Syntax, Type System and Semantics: International Standard," (2002), http://www.iso.org/iso/en/ittf/PubliclyAvailableStandards/c021573_ISO_IEC_13568_2002(E).zip.

[15] Jones, C. and J. Woodcock (eds.), *(title to be confirmed)*, Formal Aspects of Computing (2007), (to be published).

[16] *Web presentation of the Mondex case study in KIV*, URL: http://www.informatik.uni-augsburg.de/swt/projects/mondex.html.

[17] Lynch, N. A. and F. W. Vaandrager, *Forward and Backward Simulations — Part I: Untimed Systems*, Technical Report CS-R9313, C. W. I. (1993).

[18] Owre, S., J. M. Rushby and N. Shankar, *PVS: A Prototype Verification System*, in: D. Kapur, editor, *Automated Deduction - CADE-11. Proceedings*, LNAI 607, Saratoga Springs, NY, USA (1992), pp. 748–752.

[19] Reif, W., G. Schellhorn, K. Stenzel and M. Balser, *Structured Specifications and Interactive Proofs with KIV*, in: W. Bibel and P. Schmitt, editors, *Automated Deduction — A Basis for Applications*, Applied Logic Series **II: Systems and Implementation Techniques**, Kluwer Academic Publishers, Dordrecht, 1998 pp. 13–39.

[20] Schellhorn, G., *Verification of ASM Refinements Using Generalized Forward Simulation*, Journal of Universal Computer Science (J.UCS) **7** (2001), pp. 952–979, URL: http://www.jucs.org.

[21] Schellhorn, G., *ASM Refinement and Generalizations of Forward Simulation in Data Refinement: A Comparison*, Journal of Theoretical Computer Science **vol. 336, no. 2-3** (May 2005), pp. 403–435.

[22] Schellhorn, G., H. Grandy, D. Haneberg, N. Moebius and W. Reif, *A Systematic Verification Approach for Mondex Electronic Purses using ASMs*, in: U. G. J.-R. Abrial, editor, *Proceedings of the Dagstuhl Seminar on Rigorous Methods for Software Construction and Analysis*, LNCS (2007), (submitted, extended version available as [23]).

[23] Schellhorn, G., H. Grandy, D. Haneberg and W. Reif, *A Systematic Verification Approach for Mondex Electronic Purses using ASMs*, Technical Report 27, Universität Augsburg, Fak. für Informatik (2006), available at [16].

[24] Schellhorn, G., H. Grandy, D. Haneberg and W. Reif, *The Mondex Challenge: Machine Checked Proofs for an Electronic Purse*, in: J. Misra, T. Nipkow and E. Sekerinski, editors, *Proc. FM 2006*, LNCS **4085** (2006), pp. 16–31.

[25] Spivey, J., "The Z Notation: A Reference Manual," Prentice-Hall, 1992, second edition.

[26] Stepney, S., D. Cooper and J. Woodcock, *An Electronic Purse: Specification, Refinement and Proof*, Technical Report PRG-126, Oxford University Computing Laboratory (2000).

[27] The RAISE Language Group, "The RAISE Specification Language," The BCS Practitioners Series, Prentice-Hall, 1992.

[28] Weikum, G. and G. Vossen, "Transaction Processing," Morgan Kaufmann, 2002.