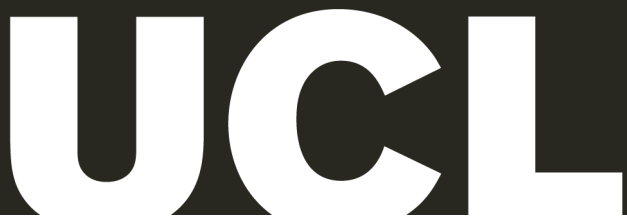
UCL DEPARTMENT OF
COMPUTER SCIENCE



Research Note
RN/15/05

Hennessy-Milner Completeness in Transition Systems with Synchronous Concurrent Composition

8th December 2015

Gabrielle Anderson

James Brotherston

David Pym

Abstract

We consider the problem of obtaining Hennessy-Milner soundness and completeness --- the coincidence of logical equivalence and bisimilarity --- in the setting of transition systems with synchronous concurrent composition. Starting from a richly expressive modal logic, motivated by resource semantics and distributed systems modelling, including both additive and multiplicative propositional connectives and also additive and multiplicative action modalities, as well as certain first-order quantifiers, we establish sufficient conditions for Hennessy--Milner soundness and completeness to hold. We develop two examples in detail. First, using the propositional part of the logic, we consider a calculus of resources and processes, explaining how the semantics may be refined to give a familiar equational theory. Second, employing a first-order, arithmetic theory, we consider a calculus with utilities that is able to express optimality and equilibria in resource allocation.

Hennessy-Milner Completeness in Transition Systems with Synchronous Concurrent Composition

Gabrielle Anderson, James Brotherston, and David Pym

University College London

Abstract. We consider the problem of obtaining Hennessy-Milner soundness and completeness — the coincidence of logical equivalence and bisimilarity — in the setting of transition systems with synchronous concurrent composition. Starting from a richly expressive modal logic, motivated by resource semantics and distributed systems modelling, including both additive and multiplicative propositional connectives and also additive and multiplicative action modalities, as well as certain first-order quantifiers, we establish sufficient conditions for Hennessy–Milner soundness and completeness to hold. We develop two examples in detail. First, using the propositional part of the logic, we consider a calculus of resources and processes, explaining how the semantics may be refined to give a familiar equational theory. Second, employing a first-order, arithmetic theory, we consider a calculus with utilities that is able to express optimality and equilibria in resource allocation.

1 Introduction

We consider the problem of obtaining Hennessy-Milner soundness and completeness — the coincidence of logical equivalence and bisimilarity — in the setting of transition systems with synchronous concurrent composition. This declarative–operational equivalence property is an important tool for modelling methodologies based on logic and transition systems. Starting from a richly expressive modal logic, motivated by resource semantics and distributed systems modelling, including both additive and multiplicative propositional connectives and also additive and multiplicative action modalities, as well as certain first-order quantifiers, we establish sufficient conditions for Hennessy–Milner soundness and completeness to hold. We develop two examples in detail. First, using the propositional part of the logic, we consider a calculus of resources and processes, explaining how the semantics may be refined to give a familiar equational theory. Second, employing a first-order, arithmetic theory, we consider a calculus with utilities that is able to express optimality and equilibria in resource allocation.

In [20], O'Hearn and Pym introduced BI, the logic of bunched implications. BI's semantics is based on preordered partial monoids and can be interpreted as providing an account of resources in terms of their combination and comparison.

BI's theory has been developed in various settings, including [21, 13]. In [11, 10], a modal extension of BI, called MBI, was introduced as a Hennessy–Milner-style logic associated with a calculus, SCRP, of co-evolving resource–process pairs. MBI is based on BI's monoidal resource semantics.

Although a great deal of theory — as well as many rich examples and industry-strength modelling applications [9, 3, 16, 7] — can be developed for SCRP and MBI, the development presented in [11, 10] is hampered by the lack of a full Hennessy–Milner soundness and completeness theorem (henceforth 'Hennessy–Milner completeness' theorem). This weakness derives from the failure of the natural notion of bisimulation for SCRP to be a congruence, so that the soundness direction of the theorem obtains only in the absence of BI's multiplicative implication ($-\!\!*$) and the multipicative modalities (that is, action modalities whose truth is parametrized by additional, local resource) [11, 10].

Recently, in [2], Anderson and Pym have obtained a full Hennessy–Milner completeness theorem for versions of SCRP and MBI based on a revised resource semantics. In this semantics, resources are bunched, and may be combined using two combinators, $\otimes$ and $\oplus$. While $\otimes$ corresponds to the monoidal composition of BI's resource semantics, and is used to interpret concurrent composition, $\oplus$ provides a combinator corresponding to non-deterministic choice.

In this paper, we provide a more general perspective in which we start, in Section 2, from a more abstract formulation of MBI that is based on a semantics that employs a labelled transition relation on states. We also include first-order predication and quantification over term and action variables (for reasons that will become clear below). We then obtain general conditions on the transition relation under which Hennessy–Milner completeness holds.

In Section 3, we explain how the resource–process calculus of [2] — based on the combinatorially richer resource semantics described above — provides an instance of the required set-up. We also give an extended example, based on semaphores, of how the resource–process calculus and the logic MBI are used.

Enriching the combinatorial structure of the resource semantics is, however, not the only way to recover Hennessy–Milner completeness. By working with a much weaker process structure, in which transitions between resources are labelled by actions, we can recover Hennessy–Milner completeness whilst introducing new features to the semantics. We illustrate this direction, in Section 4, by introducing a simple notion of utility (see, for example, [24]) to BI's (and MBI's) elementary resource semantics. This development requires the first-order structure mentioned above. Specifically, we introduce strategies and payoffs, incorporating them into the definition of bisimulation for resource transitions, while obtaining the conditions determined in Section 2 for Hennessy–Milner completeness to hold. We give an extended example of the the use of resource semantics with utility by using MBI to reason about the Prisoner's Dilemma problem and the notion of best response (see [24] for an introduction to these topics).

Finally, in Section 5, we discuss some directions for research further exploring the relationship between resources, dynamics, and logic.

2 A modal logic of resources and processes

In this section, we define an expressive modal logic, MBI, for expressing properties of transition systems with concurrency. We define a semantics for MBI in terms of a transition relation with a notion of ('concurrent') composition on its states, and its corresponding bisimulation relation. Our key technical result is that, for any model of MBI in which (a) bisimulation is a congruence with respect to concurrent composition, (b) any state can only evolve in finitely many ways under a given action, and (c) all predicates are closed under bisimulation, we have full Hennessy–Milner completeness for the logic: that is, two states are bisimilar if and only if they satisfy the same MBI formulas. We write vector notation to abbreviate tuples.

We assume a two-sorted first order language Σ, building standard terms t, u, etc., from standard variables x, y, z, etc., and *action terms*, denoted w, w', etc., built from *action variables* α, β, etc., that contains constants for all the actions $a \in \mathbf{Act}$. The predicate symbols of the language, however, may be applied to standard terms *only*.

Definition 2.1 (MBI-model). *A* model *M of MBI, together with a valuation ρ of variables, interprets standard terms in a carrier set $\mathcal{D}$ and action terms in a set $\mathcal{A}$ of* actions*, in the manner familiar from first-order logic. We write t^M for the interpretation of the term t in model M (extended pointwise to tuples of terms). Models contain the following elements:*

- *a set $\mathcal{S}$ (of* states*), equipped with partial binary composition $\circ : \mathcal{S} \times \mathcal{S} \rightharpoonup \mathcal{S}$ and distinguished element $s_e \in \mathcal{S}$;*
- *an interpretation $\mathrm{p}^M \subseteq \mathcal{S} \times \mathcal{D}^k$, for each predicate symbol* p *of arity k; and*
- *an action-indexed* transition relation *on states, $\xrightarrow{a} : \mathcal{S} \times \mathcal{S}$, where $a \in \mathcal{A}$.*

For the remainder of this section, we assume a fixed MBI-model. If $s, s' \in \mathcal{S}$ then we write $s \circ s' \downarrow$ to mean that $s \circ s'$ is defined. We write $r \to s$ if there exists some a such that $r \xrightarrow{a} s$, $\to^*$ for the reflexive, transitive closure of $\to$, and $\to^+$ for the transitive closure of $\to$.

The transition relation $\xrightarrow{a}$ induces a notion of *bisimulation* between states in the standard way.

Definition 2.2 (Bisimulation). *Define $\sim$ to be the largest symmetric binary relation on states such that, whenever $s_1 \sim s_2$ and $s_1 \xrightarrow{a} s_1'$, then there exists s_2' such that $s_2 \xrightarrow{a} s_2'$ and $s_1' \sim s_2'$.*

We will examine two different concrete models of MBI in Sections 3 and 4, the first based on a calculus for processes equipped with resources, the second on a calculus for actions on resources with an associated notion of utility.

Definition 2.3 (MBI-formulae). Formulae *of MBI are given by the following grammar, where P ranges over predicate symbols, w over action terms and $\boldsymbol{t}$ over tuples of standard terms (of appropriate length):*

$$\phi ::= \mathrm{p}\boldsymbol{t} \mid \bot \mid \phi \to \phi \mid \langle w \rangle \phi \mid [w]\phi \mid I \mid \phi * \phi \mid \phi \mathrel{-\!\!*} \phi \mid \langle w \rangle_\nu \phi \mid [w]_\nu \phi \mid \exists x.\phi \mid \exists \alpha.\phi$$

$$
\begin{array}{lll}
s \models_\rho \bot & \text{never} & \\
s \models_\rho \mathrm{p}\boldsymbol{t} & \text{iff} & (s, \rho(\boldsymbol{t})) \in \mathrm{p}^M \\
s \models_\rho \phi_1 \to \phi_2 & \text{iff} & s \models_\rho \phi_1 \text{ implies } s \models_\rho \phi_2 \\
s \models_\rho \langle w \rangle \phi & \text{iff} & \exists s'.\ s \xrightarrow{\rho(w)} s' \text{ and } s' \models_\rho \phi \\
s \models_\rho [w] \phi & \text{iff} & \forall s'.\ \text{if } s \xrightarrow{\rho(w)} s',\ \text{then } s' \models_\rho \phi \\
s \models_\rho I & \text{iff} & s \sim s_e \\
s \models_\rho \phi_1 * \phi_2 & \text{iff} & \exists s_1, s_2.\ s \sim s_1 \circ s_2 \text{ and } s_1 \models_\rho \phi_1 \text{ and } s_2 \models_\rho \phi_2 \\
s \models_\rho \phi_1 \mathrel{-\!\!*} \phi_2 & \text{iff} & \forall s.\ \text{if } s' \models_\rho \phi_1 \text{ and } s \circ s' \downarrow,\ \text{then } s \circ s' \models_\rho \phi_2 \\
s \models_\rho \langle w \rangle_\nu \phi & \text{iff} & \exists s', s''.\ s \circ s' \downarrow \text{ and } s \circ s' \xrightarrow{\rho(w)} s'' \text{ and } s'' \models_\rho \phi \\
s \models_\rho [w]_\nu \phi & \text{iff} & \forall s', s''.\ \text{if } s \circ s' \downarrow \text{ and } s \circ s' \xrightarrow{\rho(w)} s'',\ \text{then } s'' \models_\rho \phi \\
s \models_\rho \exists x.\phi & \text{iff} & s \models_{\rho[x:=d]} \phi \text{ for some } d \in \mathcal{D} \\
s \models_\rho \exists \alpha.\phi & \text{iff} & s \models_{\rho[\alpha:=a]} \phi \text{ for some } a \in \mathcal{A}
\end{array}
$$

Fig. 1. Satisfaction relation for MBI

Intuitively, the $\langle w \rangle$ and $[w]$ modalities are the familiar 'possibly' and 'necessarily' action modalities (cf. Hennessy-Milner logics for process algebras such as CCS), while the connectives I, $*$, and $\mathrel{-\!\!*}$ are respectively the multiplicative conjunction, implication and unit familiar from bunched logics [13] and in particular Boolean BI [17]. The modalities $\langle w \rangle_\nu$ and $[w]_\nu$ are possibly / necessarily modalities with respect to *state addition* in a certain sense, and the quantifiers range over actions or domain values in the usual way. Negation can be defined as implication of $\bot$, and $\vee$, $\wedge$, and $\forall$ by classical duality. The modalities are all displayed for convenience.

Now we give a Kripke-style frame semantics for MBI. First, a *valuation* is a function mapping standard variables to elements of $\mathcal{D}$ and action variables to actions in $\mathcal{A}$. Valuations extend to (standard/action) terms in the usual way. We can then define the semantics of formulas ϕ via the satisfaction relation $s \models_\rho \phi$, where $s \in \mathcal{S}$ and ρ is a valuation. The definition of our satisfaction relation is given by Figure 1. In the sequel, we drop the model M or the valuation ρ, writing $s \models_\rho \phi$ or $s \models \phi$, when their definitions are obvious.

We can observe that the resource-additional modalities, $\langle w \rangle_\nu$ and $[w]_\nu$, are in fact already definable in the rest of the logic:

Proposition 2.4. *For any model of MBI, we have the logical equivalences* $\langle w \rangle_\nu \phi \dashv\models \neg(\top \mathrel{-\!\!*} \neg\langle w \rangle \phi)$ *and* $[w]_\nu \phi \dashv\models \top \mathrel{-\!\!*} [w]\phi$.

Now, we define a *logical equivalence* between states as follows.

Definition 2.5 (Logical equivalence). *Fix some model M. Then, $r \equiv_{\mathrm{MBIU}} s$ if and only if, for all valuations ρ and formulae ϕ, $r \models_{M,\rho} \phi$ if and only if $s \models_{M,\rho} \phi$.*

In order for the Hennessy-Milner completeness property to hold for our logic (see below), we shall require the following crucial properties of our models:

Predicate $\sim$-closure: If $s \sim s'$ and $(s, \boldsymbol{d}) \in \mathrm{p}^M$, then $(s', \boldsymbol{d}) \in \mathrm{p}^M$;
Image-finiteness: For any $s \in \mathcal{S}$ and $a \in \mathcal{A}$ there are only finitely many s' with $s \xrightarrow{a} s'$;
Congruence: If $s_1 \sim s_1'$ and $s_2 \sim s_2$' and $s_1 \circ s_2 \downarrow$, then $s_1' \circ s_2' \downarrow$ and $s_1 \circ s_2 \sim s_1' \circ s_2'$.

With these properties in place, we can prove the Hennessy–Milner completeness theorem.

Theorem 2.6 (Hennessy-Milner completeness for MBI). *For any states s_1 and s_2, we have $s_1 \equiv_{\mathrm{MBI}} s_2$ if and only if $s_1 \sim s_2$.*

Proof. Straightforward. The *if* direction of the equivalence relies upon the predicate $\sim$-closure and congruence properties above, while the *only if* direction relies upon image-finiteness. □

3 Example: A calculus of resources and processes

One modelling approach, which might be expected be an example of our approach, is that based on the resource-process calculi given in [11, 10]. These calculi consist of two components: resources, which describe objects that can be created, moved, and consumed; and processes, which describe the dynamics of systems, and have various algebraic properties. Each component has a notion of composition, and so resource-process pairs have the obvious composition pairwise on the components. An action-indexed transition system can be defined in terms of a structural operational semantics over the structure of processes, so that resources and processes (i.e., the state) co-evolve: $R, E \xrightarrow{a} R', E'$. In this set-up, MBI's worlds are states, so that we work with a satisfaction relation of the form $R, E \models \phi$.

Unfortunately, in such calculi (for example, in [11, 10]), bisimulation fails to be a congruence for concurrent composition. As a result, the soundness direction of the Hennessy–Milner property holds only for fragments of the logic that exclude the multiplicative implication ($-\!\!*$) and the multiplicative modalities (here $\langle a \rangle_\nu$ and $[a]_\nu$). Bisimulation fails to be a congruence because of the way in which the resource semantics interacts with the resource-process operational semantics. Resources can be viewed as being 'capabilities', which enable behaviour in the process components of the pairs. When performing concurrent composition, these 'capabilities' can be exchanged between the process components of the pairs, enabling different behaviour in different compositions. This clearly violates the required congruence property.

In order to resolve this issue, and gain the congruence property, we change the resource semantics to ensure that 'capabilities' cannot be exchanged between process components in the operational semantics. We introduce additional structure to the resource model, beyond that in [11, 10]. The key property is injectivity

of concurrent composition, which enables us to prove the requisite congruence property of concurrent composition. We then describe how actions modify resources — that is, the resource semantics — and introduce the various definitions required to describe processes. We specify a structural operational semantics for resource-process pairs, and then state the required congruence result.

Let $\mathbf{R}$ be a set of resources, equipped with an 'empty' element $e \in \mathbf{R}$. We write R, S, etc. to denote resources. We consider unique (partial) concurrent composition of, and non-deterministic choice between, resources. In [22, 20, 11, 10], and other works in the relevant logic tradition, *bunches* are trees with leaves labelled by atomic resources, and internal nodes labelled by either $\oplus$ or $\otimes$. We implement bunching through the use of two injective functions; a resource is a node of a particular type if there exists some (unique) pair of resources that are mapped to the initial resource by the relevant function.

Definition 3.1 (Resource model). *A* resource model $(\mathbf{R}, e, \otimes, \oplus)$ *is a structure consisting of a set of resources* $\mathbf{R}$ *with a distinguished 'empty' resource* $e \in \mathbf{R}$*, and two injective, partial functions* $\otimes, \oplus : \mathbf{R} \times \mathbf{R} \rightharpoonup \mathbf{R}$*.*

In the sequel, when we write an expression of the form $R \otimes S$ or $R \oplus S$, we assume that the result of the application of the partial function to its arguments is defined. Actions correspond to the events of a system. In resource-process algebra as set up in [11, 10], actions are used to determine how resources evolve. This necessitates a relationship between the structure of actions and the structure of resources. To obtain an analogous relationship in our setting (formally stated in Definition 3.3), we also require action composition to be injective.

Definition 3.2 (Actions). *An* action model $(\boldsymbol{A}, \cdot, 1)$ *is a structure consisting of a set of actions with a distinguished 'unit' action* $1 \in \boldsymbol{A}$*, and an injective, total function* $\cdot$*.*

Note that we do not require that 1 be a unit for $\cdot$, so that $\mathbf{A}$ is not a monoid. Let ab denote $a \cdot b$. An atomic action is an action a such that there do not exist actions a_1 and a_2 such that $a = a_1 \cdot a_2$. The semantics of resources is then given by a functional relationship from action-resource pairs to resources.

Definition 3.3 (Modification functions). *A partial function* $\mu : \boldsymbol{A} \times \mathbf{R} \rightharpoonup \mathbf{R}$ *is a modification function if, for all resources* $R, S \in \mathbf{R}$ *and actions* $a, b \in \mathbf{Act}$*:*

- *If* $\mu(a, R)$*,* $\mu(b, S)$*,* $R \otimes S \downarrow$*, then* $\mu(ab, R \otimes S) = \mu(a, R) \otimes \mu(b, S)$*;*
- $\mu(1, R) = R$*.* □

Modification functions are homomorphisms with respect to the concurrent product structure of resource bunches. As a result, we cannot use the modification function to 'move' resources from one side of a concurrent product to another (such a move corresponds to changing the process to which the resources are allocated, for example, passing an object from producer to consumer). Using a modification function, we can only add or remove resources to each side of a product independently of what is on the other side of the concurrent product.

As we cannot use a modification function for redistribution of resources, instead, we make use of *redistribution functions*. In Figure 2, the rules for the operational semantics of sequential composition are

$$\frac{R, E \xrightarrow{a} R', E' \rightarrow \quad \gamma \in \Gamma}{R, E :_\gamma F \xrightarrow{a} R', E' :_\gamma F} \textsc{PrefixOne} \qquad \frac{R, E \xrightarrow{a} R', E' \nrightarrow \quad \gamma \in \Gamma}{R, E :_\gamma F \xrightarrow{a} \gamma(R'), F} \textsc{PrefixTwo}$$

The resource-process pair $R, E :_\gamma F$ consists of a resource bunch and a sequential composition. The sequential composition consists of two processes, E and F, and a redistribution function γ. If the prefix E can evolve with the resources R to a non-blocked state, then the sequential composition evolves similarly (the PrefixOne rule). If the prefix E can evolve with the resources R to a blocked state, then the redistribution function is applied to the resulting resources R', and the pair that consists of the redistributed resources and the suffix, $\gamma(R'), F$, is the result of the transition (the PrefixTwo rule). The redistribution function is applied to the resources so that the structure of the resulting resources will match the structure of the suffix process. Redistribution functions are total so that the evolution of a sequential composition can only be blocked by the behaviour of the prefixing process, not the redistribution of resources.

Definition 3.4 (Redistribution functions). *A redistribution function is a (partial) function $\gamma : \mathbf{R} \rightharpoonup \mathbf{R}$. Let there be a set of redistribution functions Γ whose elements are written γ, γ', etc..*

Redistribution functions are total so that the evolution of a sequential composition can only be blocked by the behaviour of the prefixing process, not the redistribution of resources. From a modelling perspective, we argue that the use of redistribution functions encourages good discipline with respect to making decisions about how resources are allocated to processes within a system.

In classical process calculi, *restriction* is used to ensure that certain behaviour is only visible, or accessible, in certain parts of a system. A similar feature can be incorporated into resource–process modelling [11]. If a resource-process pair is allocated additional resources, it may be able to perform additional behaviour. The *hiding* operator on processes associates additional resources with the process to which it is applied. If a resource–process pair is allocated additional resources, it may be able to perform additional actions. This behaviour must then be restricted, however; only actions that could be performed without the additional resources must be visible beyond the process where the hidden resources are available. First, we define a notion of action containment, so that we can formalize the notion of 'additional behaviour'.

Definition 3.5 (Action-containment order). *We define $\leqslant$ to be the least reflexive-transitive relation on actions such that $1 \leqslant \alpha$ for any atomic action α, and if $a \leqslant a'$ and $b \leqslant b'$ then $a \cdot b \leqslant a' \cdot b'$.*

Then, we define hiding functions on actions and resources. In Figure 2, the rule for the operational semantics of hiding functions is

$$\frac{h(R), E \xrightarrow{a} h(R'), E' \quad h \in \mathbf{H}}{R, \nu h.E \xrightarrow{\nu h.a} R', \nu h.E'} \textsc{Hide}.$$

A resource-process pair $R, \nu\, h.E$ evolves by stripping the hiding operator $\nu\, h.$ from the process component and applying the hiding function h to the resource component, resulting in the resource-process pair $h(R), E$. Following the evolution of the transformed state, the resulting pair $h(R'), E'$ is modified by applying the inverse of the hiding function to the resource component and adding the hiding operator to the process component, resulting in the resource-process pair $R', \nu\, h.E'$. To ensure that a hiding function and its inverse can be uniquely applied, hiding functions on resources are bijections.

Definition 3.6 (Hiding functions). *Let* $(\mathbf{R}, e, \otimes, \oplus)$ *be a resource model and* μ *be a modification function. A function* $h : \mathbf{R} \to \mathbf{R}$ *on a resource model is a hiding function if it is a bijection. Let there be a set of hiding functions* $\mathbf{H}$ *whose elements are written* h, h', *etc.. Define* $A : (\mathbf{R} \to \mathbf{R}) \to \mathbf{Act} \to \mathcal{P}(\mathbf{Act})$

$$A(h, a) = \{b \leqslant a \mid \textit{for all } R, S \in \mathbf{R}, \mu(a, h(R)) = h(S) \textit{ implies } \mu(b, R) = S\}\ .$$

Then, a hiding function on actions $\nu : (\mathbf{R} \to \mathbf{R}) \to \mathbf{Act} \to \mathbf{Act}$ *is defined as*

$$\nu h.a = \begin{cases} \sup(A(h, a)) & \textit{if } \sup(A(h, a)) \textit{ is defined and unique} \\ 1 & \textit{otherwise.} \end{cases}$$

Definition 3.7 (Processes). *The set* **Proc** *of* processes *is given by the following grammar:*

$$E ::= \mathbf{0} \mid X \mid a \mid E + E \mid E \times E \mid E :_\gamma E \mid \nu h.E \mid \mathit{fix}\, X.E.$$

where, $\mathbf{0}$ *is the zero process,* X *is a process variable,* a *is an action,* $h \in \mathbf{H}$ *is a hiding function, and* $\gamma \in \Gamma$ *is a redistribution function.*

We write E, F, etc. to denote processes. The process $\mathbf{1}$, which performs the action 1 infinitely, is denoted as $\mu X.1 :_{id} X$. The process structure broadly follows that of ACP [4]. Thus $E + F$ is a *sum*, $E \times F$ is a *synchronous product*, and $\mathrm{fix}\, X.E$ is a *fixed point*. The term $\nu h.E$ is a *hiding* process, as in [11, 10]. The term $E :_\gamma F$ is an *annotated sequential composition*. The fix operator binds occurrences of process variables within processes. Here, we only consider process expressions that are *closed*, in that they contain no free variables. We define a *state* to be a pair consisting of a resource and a (closed) process, and write $\mathbf{State} = \mathbf{R} \times \mathbf{Proc}$ for the set of all states.

We make use of an empty term language. Let the action term language be formed according to the grammar $w ::= a \mid \alpha \mid w \diamond w$, where a is a constant denoting the action a, and there exists a constant a for each action $a \in \mathbf{Act}$.

$$\frac{}{R, a \xrightarrow{a} \mu(a, R), \mathbf{0}}\ \textsc{Act} \qquad \frac{R_i, E_i \xrightarrow{a} R_i', E_i'}{R_1 \oplus R_2, E_1 + E_2 \xrightarrow{a} R_i', E_i'}\ \textsc{Sum},\ i \in \{1, 2\}$$

$$\frac{R_1, E_1 \xrightarrow{a_1} R_1', E_1' \quad R_2, E_2 \xrightarrow{a_2} R_2', E_2'}{R_1 \otimes R_2, E_1 \times E_2 \xrightarrow{a_1 \cdot a_2} R_1' \otimes R_2', E_1' \times E_2'}\ \textsc{Prod}$$

$$\frac{R, E \xrightarrow{a} R', E' \quad \gamma \in \Gamma}{R, E :_\gamma F \xrightarrow{a} R', E' :_\gamma F}\ \textsc{PrefixOne} \qquad \frac{R, E \twoheadrightarrow \quad \gamma(R), F \xrightarrow{a} R', F' \quad \gamma \in \Gamma}{R, E :_\gamma F \xrightarrow{a} R', F'}\ \textsc{PrefixTwo}$$

$$\frac{h(R), E \xrightarrow{a} h(R'), E' \quad h \in \mathbf{H}}{R, \nu h.E \xrightarrow{\nu h.a} R', \nu h.E'}\ \textsc{Hide} \qquad \frac{R, E[\text{fix } X.E/X] \xrightarrow{a} R', E'}{R, \text{fix}\, X.E \xrightarrow{a} R'.E'}\ FV(E) \subseteq \{X\}\ \textsc{Rec}$$

Fig. 2. Operational Semantics

Valuations map action constants to their obvious actions and $\diamond$ to action composition $\cdot$.

In order to obtain the Hennessy–Milner completeness result stated in Section 2, we must show that our resource-process calculi are instances of the class of systems considered in that section. States are resource-process pairs $\mathbf{R} \times \mathbf{Proc}$, and the distinguished state is $(e, \mathbf{1})$. Concurrent composition of states maps $\circ : ((R_1, E_1), (R_2, E_2)) \mapsto (R_1 \otimes R_2, E_1 \times E_2)$ if and only if $R_1 \otimes R_2$ is defined. The action-indexed transition relation on states is defined recursively using the derivation rules in Figure 2.

Then, all that remains is to show that bisimulation is a congruence with respect to the composition $\circ$. Note that, when composing states, it is important to take account of the partiality of the resource model. As a result, when composing states concurrently, we shall require the following $\sim$*-resource closure* property of our calculi: supposing $R_1, E_1 \sim S_1, F_1$ and $R_2, E_2 \sim S_2, F_2$, we have that $R_1 \otimes R_2$ (respectively, $R_1 \oplus R_2$) is defined if and only if $S_1 \otimes S_2$ (respectively, $S_1 \oplus S_2$) is defined. From now on, all calculi are assumed to be $\sim$-resource-closed. As an additional result, when composing states sequentially, we require the following γ*-sequence closure* property of our calculi: a state R, E and a bunched resource S are γ-sequence compatible if, for all states R', E' that can be reached by the transition system, $R, E \rightarrow^+ R', E'$, we have that $R', E' \twoheadrightarrow$ implies $\gamma(R') = S$.

We can now obtain the key property: that bisimulation is a congruence, i.e. an equivalence relation that is respected by the state constructors (excepting the fixed point constructor). Note that injectivity concurrent composition (the first of the two properties, above) is all that is required to obtain the congruence property for concurrent composition, which is all that is required to obtain the results in in Section 2.

$$\dfrac{\dfrac{\dfrac{\mu(a,s)=s}{s,a \xrightarrow{a} s,\mathbf{0}}}{(s\oplus s),(1+a)\xrightarrow{a} s,\mathbf{0}} \quad \dfrac{\dfrac{\mu(1,e)=e}{e,1\xrightarrow{1} e,\mathbf{0}}}{(e\oplus e),(1+b)\xrightarrow{1} e,\mathbf{0}}}{(s\oplus s)\otimes(e\oplus e),(1+a)\times(1+b)\xrightarrow{a\cdot 1} s\otimes e,\mathbf{0}\times\mathbf{0}}}{R\oplus S,E+E\xrightarrow{a\cdot 1} s\otimes e,\mathbf{0}\times\mathbf{0}}$$

Fig. 3. First process accesses the semaphore

$$\dfrac{\dfrac{\dfrac{\mu(1,e)=e}{e,1\xrightarrow{1} e,\mathbf{0}}}{(e\oplus e),(1+a)\xrightarrow{1} e,\mathbf{0}} \quad \dfrac{\dfrac{\mu(b,s)=s}{s,b\xrightarrow{b} s,\mathbf{0}}}{(s\oplus s),(1+b)\xrightarrow{b} s,\mathbf{0}}}{(e\oplus e)\otimes(s\oplus s),(1+a)\times(1+b)\xrightarrow{1\cdot b} e\otimes s,\mathbf{0}\times\mathbf{0}}}{R\oplus S,E+E\xrightarrow{1\cdot b} e\otimes s,\mathbf{0}\times\mathbf{0}}$$

Fig. 4. Second process accesses the semaphore

Theorem 3.8 (Bisimulation congruence). *The relation $\sim$ is a congruence for concurrent, non-deterministic, and sequential composition, and hiding:*

- *if $R_i, E_i \sim S_i, F_i$ for $i \in \{1,2\}$ and $R_1 \otimes R_2 \downarrow$, then $S_1 \otimes S_2 \downarrow$ and $R_1 \otimes R_2, E_1 \otimes E_2 \sim S_1 \otimes S_2, F_1 \otimes F_2$;*
- *if $R_i, E_i \sim S_i, F_i$ for $i \in \{1,2\}$ and $R_1 \oplus R_2 \downarrow$, then $S_1 \oplus S_2 \downarrow$ and $R_1 \oplus R_2, E_1 + E_2 \sim S_1 \oplus S_2, F_1 + F_2$;*
- *if h is a hiding function with $h(R), E \sim h(S), F$, then $R, \nu h.E \sim S, \nu h.E$;*
- *if R_1, E_1, and R_2 are γ-sequence compatible, and S_1, F_1, and S_2 are γ'-sequence compatible, and $R_i, E_i \sim S_i, F_i$ for $i \in \{1,2\}$, then $R_1, E_1 :_\gamma E_2 \sim S_1, F_1 :_{\gamma'} F_2$.*

Hence, our resource-process calculi are instances of the class of systems considered in Section 2, and have the Hennessy-Milner completeness property.

The SCRP framework has been used to underpin significant industrial modelling [9, 10, 16, 6]. In order to be able to use CBRP as a replacement for SCRP, we should be able to embed the latter soundly into the former. It is indeed possible to do this; the approach is described formally in [1]. In order to reason equationally about processes, it is also useful to establish various algebraic properties concerning concurrent composition and choice. Notable standard algebraic properties of process calculi are commutativity and associativity of concurrent composition, that is, $R \otimes S, E \times F \sim S \otimes R, F \times E$ and $R \otimes (S \otimes T), E \times (F \times G) \sim (R \otimes S) \otimes T, (E \times F) \times G$. For the notion of bisimulation in Definition 2.2, these properties do not generally hold. However, we can recover these and other algebraic properties by quotienting bisimilarity $\sim$ with respect to a natural notion of equivalence between *actions*. The technical details, which are straightforward, can likewise be found in [1].

We conclude this section with a fully fledged modelling example of how to model an unbounded series of accesses to a semaphore by concurrent processes, in a resource-process calculus.

Example 3.9 (Semaphores). Consider a typical contested resource, a semaphore. Semaphores are objects that should be 'held" (or 'used') by at most one process in a concurrent composition of processes. In this section, we describe how to model two concurrent processes competing for the use of a semaphore in a resource-process setting.

We model the resource aspect of the scenario as follows. Let the resource s denote the semaphore, and e denote the empty resource. Suppose that U and V are (arbitrary) resources. We use (U, V) pairs to denote parallel compositions of resources U and V, and $(U; V)$ pairs to denote choice compositions of resources U and V. Let $U \otimes V = (U, V)$ if and only if s does not occur in both U and V, and $U \oplus V = (U; V)$.

We make particular use of the resources $R = ((s; s), (e; e))$, $S = ((e; e), (s; s))$, and $T = (R; S)$. Let the set of resources be $\mathbf{R} = \{s, e, (s, e), (e, s), R, S, T, \ldots\}$. Then, $(\mathbf{R}, e, \otimes, \oplus)$ is a resource model. Note then, that $s \otimes s$ is undefined: this models the key property of the scenario that two concurrent processes cannot both hold a copy of the semaphore.

We model the process dynamics aspect of the scenario as follows. We use two atomic actions, a and b, both of which require access to a semaphore, s, to be performed. We differentiate between the a and b actions to help make clear which process is accessing the semaphore at any given point. Let $\mathbf{Act} = \{a, b\}$ be set of atomic resources. Let μ be the least modification function such that $\mu(a, s) = s$ and $\mu(b, s) = s$. The process $E = (1 + a) \times (1 + b)$ denotes a system where two concurrent processes each attempt to access the semaphore (through actions a and b respectively). The resource R denotes the scenario where the semaphore is allocated to the first process, and S where it is allocated to the second process. The resource T then denotes the scenario where the semaphore may be allocated to either of the processes, but not to both. The state $T, E + E$ can either evolve through use of the resource R (with process E), or through the use of the resource S (with process E). In the first case, the first process can access the semaphore, but the second process can only tick (Figure 3). In the second case, the converse is true (Figure 4).

It should not be possible for an action a and an action b to be performed concurrently. This property can be represented formally by the logical formula $\phi_1 = \neg((\langle a \rangle \top) * (\langle b \rangle \top))$. In this example, $R \oplus S, E + E \vDash \phi_1$ can only hold if $R \oplus S, E + E \vDash ((\langle a \rangle \top) * (\langle b \rangle \top))$ *doesn't* hold. That is only the case if, for all R_1, E_1, R_2, E_2, such that $R_1 \otimes R_2, E_1 \times E_2 \sim R \oplus S, E + E$, either $R_1, E_1 \vDash (\langle a \rangle \top)$ doesn't hold or $R_2, E_2 \vDash (\langle b \rangle \top)$ doesn't hold. As the semaphore s is required to perform both the a and the b action, and $R_1 \otimes R_2$ is undefined when s is in both R_1 and R_2, there are no R_1, E_1, R_2, E_2, such that $R_1 \otimes R_2, E_1 \times E_2 \sim R \oplus S, E + E$ and both $R_1, E_1 \vDash (\langle a \rangle \top)$ and $R_2, E_2 \vDash (\langle b \rangle \top)$ hold. Hence, $R \oplus S, E + E \vDash \phi_1$ holds.

It is, however, possible for each of the actions to occur separately. There properties can be represented formally by the logical formulae $\phi_2 = ((\langle a \rangle \top) * (\langle 1 \rangle \top))$ and $\phi_3 = ((\langle 1 \rangle \top) * (\langle b \rangle \top))$. In this example, $R, E \vDash \phi_2$, as $(s; s), 1 + a \vDash \langle a \rangle \top$ and $(e; e), 1 + b \vDash \langle 1 \rangle \top$, and $S, E \vDash \phi_3$, as $(e; e), 1 + a \vDash \langle 1 \rangle \top$ and $(e; e), 1 + b \vDash \langle b \rangle \top$.

Furthermore, we cannot compose a resource-process pair that can perform a b action onto one that can perform an a action. This property can be represented formally by the logical formula $\phi_4 = \big((\langle a \rangle \top) \Rightarrow ((\langle b \rangle \top) \mathrel{-\!\!*} \bot)\big)$. In this example, $R, E \vDash \langle a \rangle \top$, as $R, E \xrightarrow{a}$. Hence, in order to show that $R, E \vDash \phi_4$, we have to

show that, for all U, F such that $U, F \vDash \langle b \rangle \top$ and $R \otimes U \downarrow$, $R \otimes U, E \times F \vDash \bot$. As there is no state that satisfies $\bot$, we must have that there are no such U and F. As the semaphore s is required to perform both the a and the b action, and $R_1 \otimes R_2$ is undefined when s is in both R_1 and R_2, for any U, F such that $U, F \xrightarrow{b}$, then $R \otimes U \uparrow$. Hence we have that $R, E \vDash \phi_4$. We can alternatively express this concept as $(\langle a \rangle \top) \Rightarrow ([ab]_\nu \bot)$, which denotes that it is not possible to concurrently compose any resource-process onto one that can perform an a action, such that the composed state can perform an ab action.

Although we have not made use of either redistribution functions or hiding functions in the examples of this paper, they are nonetheless a significant part of the resource–process modelling framework. The use of resources as tokens to enable behaviour in the process component of a state is central to modelling blocking behaviour in distributed systems. Without an opportunity to reallocate resources from one process to another, however, there is no way to model synchronization. Using redistribution functions, alongside concurrent and nondeterministic composition, we can model a wide variety of synchronization examples, such as mutual exclusion, joint access control, producer–consumer, and weak memory models [1].

In order to aid compartmentalization of implementation details, and hence with the scalability of cognitive complexity of modelling, hiding can be used to ensure that certain behaviour is only visible in certain parts of a system. In order to handle the additional structure of resources introduced above, hiding functions generalize the concurrent-composition approach taken to hiding in [11]. Examples of how to make use of this generalization — for example, in scenarios where distributivity of product over choice is considered — can be found in [1].

4 Example: A calculus of resources with utility

In the previous section, we added extra structure to the resource semantics in order to obtain the required congruence property of concurrent composition in resource-process modelling and so obtain Hennessy–Milner completeness.

We can, however, also recover Hennessy–Milner completeness by working with a much weaker notion of process, in which transitions between resources are labelled by actions, using the modification function on resources to provide an elementary dynamics. In this modelling approach, we do not require the additional combinatorial structure over resources that is used in the previous section, but introduce the additional concept of utility, a key tool in reasoning about notions of optimality in distributed systems. Reasoning about this notion of utility in MBI makes essential use of the first-order structure explained in Section 2.

We begin with the notion of resource from Boolean BI, which can be seen as liberalizing the combinatorial structure taken in Section 3 in that it does not consider choices between resources, nor does it require the concurrent composition to be injective.

Definition 4.1 (Resource monoid). *A resource monoid is a monoid* $\mathbf{R} = (\mathbf{R}, \circ, e)$ *with carrier set* $\mathbf{R}$*, commutative partial binary operation* $\circ : \mathbf{R} \times \mathbf{R} \rightharpoonup \mathbf{R}$*, and unit* $e \in \mathbf{R}$*.*

In the sequel, when we write an expression of the form $R \circ S$, we assume that the result of the application of the partial function to its arguments is defined. Note that this is essentially the same as resource models in Definition 3.1, but without the additive structure and the injectivity requirement.

Let $\mathbf{A}$ be a commutative monoid of actions, freely generated from a set of atomic actions, with operation $\cdot$ and unit 1. The actions correspond to the events of the system. Let ab denote $a \cdot b$. The dynamics of the system is then given by the modification function, which describes how actions transform resources.

Definition 4.2 (Modification function). *A modification function is a partial function* $\mu : \boldsymbol{A} \times \mathbf{R} \rightharpoonup \mathbf{R}$ *such that, for all resources* $R, S \in \mathbf{R}$ *and actions* $a, b, c \in \boldsymbol{A}$*:*

- *if* $\mu(a, R) \downarrow$*,* $\mu(b, S) \downarrow$*, and* $R \circ S \downarrow$*, then* $\mu(ab, R \circ S) = \mu(a, R) \circ \mu(b, S)$*;*
- $\mu(1, R) = R$*;*
- *if* $R \circ S \downarrow$ *and* $\mu(c, R \circ S) \downarrow$*, then there exist* $a, b \in \boldsymbol{A}$ *such that* $c = ab$*,* $\mu(a, R) \downarrow$*, and* $\mu(b, S) \downarrow$*.*

Note that this is essentially the same as Definition 3.3, but with one additional property, which plays a similar role to the Prod rule in Section 3 for determining the behaviour of a concurrent composition of states.

From a resource monoid, action monoid, and modification function, we derive a transition relation. If the modification function is defined for an action a on a resource R, and $\mu(a, R) = S$, then we say that there exists a transition $R \xrightarrow{a} S$, that S is a successor of R, and that action a is defined on resource R. The notion of bisimulation in Definition 2.2 is immediately applicable to resources. In this simple setting, bisimulation equivalence is the same as trace equivalence. Following Section 3, we require the following $\sim$*-closure* property: if $R_1 \sim S_1$, $R_2 \sim S_2$, and $R_1 \circ R_2 \downarrow$, then $S_1 \circ S_2 \downarrow$. From now on, all resource models are assumed to be $\sim$-closed.

In order to obtain the Hennessy–Milner completeness result stated in Section 2, we must show that our resource models are instances of the class of systems considered in that section. States, concurrent composition, and the transition relation are as defined above. Then, all that remains is to show that bisimulation is a congruence with respect to the composition $\circ$, and that the interpretation of the 'payoff' predicate, which we use in expressing optimality properties, is interpretation-$\sim$-closed. First, we present the congruence result.

Theorem 4.3 (Bisimulation congruence). *The relation* $\sim$ *on resources is a congruence for the operation* $\circ$*: if* $R_1 \sim S_1$*,* $R_2 \sim S_2$*, and* $R_1 \circ R_2 \downarrow$*, then* $S_1 \circ S_2 \downarrow$ *and* $R_1 \circ R_2 \sim S_1 \circ S_2$*.*

In order to reason about optimality properties of states and actions, we require a way to assign a value or payoff to states and actions.

Definition 4.4 (Action payoff function). *An action payoff function is a partial function* $v : \mathbf{Act} \rightharpoonup \mathbb{Q}$ *s.t.* $v(1) = 0$ *and, for all* $a, b \in \boldsymbol{A}$*, if* $v(a)$ *and* $v(b)$ *are defined, then* $v(ab) = v(a) + v(b)$*.*

Note that it is possible to have that $v(ab)$ is defined, but that $v(a)$ and $v(b)$ are not defined (cf. Example 4.8).

The transition systems generated by our resource semantics can be nondeterministic, in the sense that multiple actions can be defined on a given resource. In order to extend the notion of payoff to resources, we determine, at each resource, a unique action to be performed

Definition 4.5 (Strategies). *A strategy is a total function* $\sigma : \mathbf{R} \to \boldsymbol{A}$ *such that, for all resources* $R, S \in \mathbf{R}$*,* $\mu(\sigma(R), R)$ *is defined, and, if* $R \sim S$*, then* $\sigma(R) = \sigma(S)$*.*

Fix an action payoff function v, a strategy σ, and let δ be some rational number in the open interval $(0, 1)$. We can then straightforwardly extended the notion of preferences over actions to preferences over resources.

Definition 4.6 (Resource payoff function). *A resource payoff function is a partial function* $u_{v,\sigma,\delta} : \mathbf{R} \rightharpoonup \mathbb{Q}$ *such that*

$$u_{v,\sigma,\delta}(R) = \begin{cases} v(a) + \delta \times u_{v,\sigma,\delta}(\mu(a, R)) & \text{if } \sigma(R) = a, v(a) \downarrow, \text{ and } u_{v,\sigma,\delta}(\mu(a, R)) \downarrow \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The value that can be accumulated from actions performed at resources reachable in the future are worth less than value that can be accumulated immediately. The discount factor δ is used to discount future accumulated values. In the case that the set $\mathbf{R}$ is finite, we generate a finite set of simultaneous equations which can be solved using the methods described in [15]. Henceforth, we assume that all resource monoids have finite carrier sets. Note that bisimilar states have the same payoff.

Lemma 4.7. *If* $R \sim S$*, then for all* σ*,* δ*,* v*,* $u_{v,\sigma,\delta}(R) = u_{v,\sigma,\delta}(S)$*.*

Proof. By our assumptions, R and S both have a finite number of successor states. These states, and their relevant transition systems, can be uniquely mapped into the final coalgebra of finite and infinite sequences of actions. In particular, since $R \sim S$ we know that both are uniquely mapped to the same element of the final coalgebra (see Definition 4.5). By [18], as R and S both have a finite number of successor states, the sequence to which they are mapped is either finite or eventually periodic. Hence, the utility function can be defined over these elements of the final coalgebra in a similar fashion to how it is defined over states. To be precise both utility functions are defined as a unique coalgebra-to-algebra homomorphism (for details see [15]) and they correspond to computing the solution of a linear system of equations. As there is a unique mapping from the states to the final coalgebra, and a unique mapping from the final coalgebra to the payoff, there is a unique mapping from each of the states to the payoff, which is identical for R and S.

We interpret terms in the rational numbers. In order to relate the value of resources to terms that we can manipulate, we make use of a distinguished predicate $\mathtt{u}_\mathtt{v}(t)$, whose interpretation is that $(s, \rho(t)) \in \mathtt{u}_\mathtt{v}^M$ if and only if $u_{v,\sigma,\delta}(s) = \rho(t)$. Note that, in a given interpretation M, we fix a strategy σ and a discount factor δ. As bisimilar states have the same payoff, for fixed action payoff function, strategy, and discount factor (Lemma 4.7), the interpretation of $\mathtt{u}_\mathtt{v}(t)$ is interpretation-$\sim$-closed.

Let the action term language be formed according to the grammar $w ::= a \mid \alpha \mid w \diamond w$, where a is a constant denoting the action a. Valuations map action constants to their obvious action and $\diamond$ to action composition $\cdot$.

Let q be a term constant denoting the rational number q, and $\mathtt{v}(s)$ be a constant denoting the rational-valued payoff of an action term s according to payoff function v. Let the term language be formed according to the grammar $t ::= x \mid q \mid \mathtt{v}(s) \mid t + t \mid t \times t$. Valuations map term constants to their obvious denotations and arithmetical functions their standard definitions.

Hence, our resource modelling semantics is an instance of the class of systems considered in Section 2, and has the Hennessy-Milner completeness property.

Using this logic, it is possible to logically describe various interesting optimality properties, including Pareto optimality, best response, and Nash equilibrium [2]. For now, we conclude this section with a fully fledged modelling example of how to capture and reason about the Prisoner's Dilemma and best response in a resource semantics.

Example 4.8 (Prisoner's dilemma). Two individuals have been arrested, and are kept separately, so that they cannot collude in their decision-making. Each is offered the choice of attempting to 'defect', and give evidence against their partner, or to 'collaborate', and say nothing. If one person collaborates and the other defects, then the collaborating partner goes to jail for a long time, and the defecting partner goes free. If both defect, then they both go to jail for a moderate time. If both collaborate, then they both go to jail for a short time.

Suppose a resource monoid $(\{r_1, r_2, r_{1,2}, e\}, \circ, e)$, where $r_1 \circ r_2 = r_{1,2}$. The r_1 resource denotes a resource where the first person can make a choice, the r_2 resource denotes a resource where the second person can make a choice, and the $r_{1,2}$ resource denotes a resource where both people can make a choice at the same time. Suppose actions c_1, d_1, c_2, and d_2, where

$$\begin{gathered}\mu(c_1, r_1) = \mu(d_1, r_1) = e \qquad \mu(c_2, r_2) = \mu(d_2, r_2) = e \\ \mu(c_1c_2, r_{1,2}) = \mu(c_1d_2, r_{1,2}) = \mu(d_1c_2, r_{1,2}) = \mu(d_1d_2, r_{1,2}) = e.\end{gathered}$$

The c_1 action denotes collaboration by the first person, and the d_1 action denotes defection by the person. The c_2 and d_2 actions have the obvious denotations for the second person. We make use of the trivial strategy $\sigma(R) = 1$. The action payoff functions v_1 and v_2 for the two people are

$$\begin{array}{llll} v_1(c_1c_1) = -2 & v_1(c_1d_2) = -6 & v_1(d_1c_2) = 0 & v_1(d_1d_2) = -4 \\ v_2(c_1c_1) = -2 & v_2(c_1d_2) = 0 & v_2(d_1c_2) = -6 & v_2(d_1d_2) = -4. \end{array}$$

Hence, if the first person collaborates and the second defects, then the first person receives six years in prison (cost $v_1(c_1 d_2) = -6$), while the second receives no time in prison (cost $v_2(c_1 d_2) = 0$).

We can define a notion of *best response*. An action a is a best response for a given entity to a particular choice of action b by another entity, at a given resource, if the (former) entity has no other action c available to it such that the action cb is defined on the resource and the entity (strongly) prefers cb to ab. Formally, a is the best response to action b at resource R if

$$R \models \forall \alpha. \exists x, y. \Big(\big((\langle a \rangle \top \wedge \langle \alpha \rangle \top) * (\langle b \rangle \top) \big) \wedge \big([a \diamond b] \mathtt{u}_{\mathtt{v}}(x) \wedge [\alpha \diamond b] \mathtt{u}_{\mathtt{v}}(y) \big) \Big) \\ \rightarrow \big((v(\alpha \diamond b) + \delta \times y) \leqslant (v(a \diamond b) + \delta \times x) \big).$$

We abbreviate the formula above, denoting that a is the best response to action b for the agent whose payoff function is v, as $BR(a, b, v)$. In the prisoner's dilemma example, the best response for the first agent to the action c_2 is d_1, and $r_{1,2} \models BR(d_1, c_2, v_1)$ holds.

5 Discussion and future work

The methodology used in this paper is to start with the desired logic, determine the desired meta-theory of the logic, and then derive properties of the semantics and interpretation of the logic that are sufficient to prove the desired meta-theory of the logic. In general, this methodology provides a clear justification for design decisions in the semantics of the logic.

We have shown that a version of resource semantics in which there is a closer combinatorial match between the structure carried by resources and that carried by processes permits us to obtain the Hennessy–Milner completeness theorem for the full substructural logic, the lack thereof being a technical difficulty present in an earlier formulation of the relationship between resources and processes. As a result, our work suggests that the original ideas of resource semantics, already useful and influential in, say, separation logic, may warrant further exploration.

Some conceptual and technical issues, beyond our present scope, remain to be addressed, however. In recent work in logic [12], one of us has considered a generalization of resource semantics to admit multi-dimensional satisfaction relations of the form, for example, $w, r \models \phi$, in which $w \in W$ are taken to be Kripke worlds (ordered by $\sqsubseteq$, say) in the sense of classical modal logic and $r \in R$, where R carries monoidal structure (with composition $\circ$, say), are interpreted as resources. In this set-up, we can define, informally for now, a modality $\Diamond_s$ as

$$w, r \models \Diamond_s \phi \text{ iff there is a world } w \sqsubseteq v \text{ such that } v, r \circ s \models \phi.$$

Such a modality is highly expressive and, among other things, generalizes the usual S4 modality [8]. It may be possible to define an analogous action modality, $\langle a \rangle_{S,F}$, which generalizes our multiplicative modality $\langle a \rangle_{\nu}$:

$$R, E \models \langle a \rangle_{S,F}\ \phi \text{ iff } \exists a, R', S', E', F' \text{ such that } R \otimes S, E \otimes F \xrightarrow{a} R' \otimes S', E' \otimes F' \\ \text{and } R' \otimes S', E' \otimes F' \models \phi.$$

Note that, unlike in the previous definition, we add both a resource and a process component. We conjecture that the transition system employed in the body of this paper and the construction described above are both examples of a more general treatment of a more general multi-dimensional semantics that will have natural resource interpretations.

A further question concerns the relationship between our work and concurrent separation logic [19]. Concurrent separation logic is built upon the resource semantics of bunched logic and handles concurrent processes in the style of Hoare logic. In general, there is a close relationship between Hoare logic presentations of program logics on the one hand and modal logic presentations on the other, based on representing Hoare triples $\{P\}C\{Q\}$ as entailments of the form $P \models [C]Q$ (although this relationship is less straightforward for *fault-avoiding* interpretations of Hoare triples [23]). Nevertheless, we conjecture that our treatment of resource semantics might be used to support CSL and other concurrent phenomena too (cf. [14]), possibly including a synchronous semantics for concurrent separation logic (in contrast to Brookes' interleaving semantics [5]). Such a programme, however, awaits future investigation.

Finally, the point of view we have discussed in Section 4, related to ideas presented in [2], suggests that a more substantial exploration of ideas of agency, games, and knowledge — perhaps building on ideas in [12], with connections to epistemic game theory — may be a fruitful direction.

Acknowledgements. We are grateful to Matthew Collinson, Guy McCusker, and Alexandra Silva for their advice on writing this paper. This work has been supported by the UK EPSRC project EP/K033042/1, 'Algebra and Logic for Policy and Utility in Information Security'.

References

1. G. Anderson and D. Pym. A Calculus and Logic of Bunched Resource Processes. Accepted for a journal, subject to minor revisions, 2015. Manuscript at `http://www0.cs.ucl.ac.uk/staff/D.Pym/AndersonPymBunchedResourceProcess.pdf`.
2. G. Anderson and D. Pym. Substructural Modal Logic for Optimal Resource Allocation. In *Proc. Strategic Reasoning*, 2015.
3. Y. Beresnevichiene and D. Pym amd S. Shiu. Decision support for systems security investment. In *Network Operations and Management Symposium Workshops (NOMS Workshops), 2010 IEEE/IFIP*, pages 118–125. IEEE Xplore, 2010.
4. J. Bergstra and J. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, 37(1):77–121, 1985.
5. S. Brookes. A semantics for concurrent separation logic. *Theoretical Computer Science*, 375(1-3):227–270, 2007.
6. T. Caulfield, D. Pym, and J. Williams. Compositional Security Modelling: Structure, Economics, and Behaviour. *LNCS*, 8533:233–245, 2014.
7. Tristan Caulfield and D. Pym. Modelling and simulating systems security policy. In *Proc. 8th. SIMUTools*. ACM Digital Library, 2015.
8. B. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, 1980.
9. M. Collinson, B. Monahan, and D. Pym. Semantics for structured systems modelling and simulation. In *Proc. SIMUTools*, pages 34:1–34:10, 2010.

10. M. Collinson, B. Monahan, and D. Pym. *A Discipline of Mathematical Systems Modelling*. College Publications, 2012.
11. M. Collinson and D. Pym. Algebra and logic for resource-based systems modelling. *Mathematical Structures in Computer Science*, 19(5):959–1027, 2009.
12. J.-R. Courtault, D. Galmiche, and D. Pym. A Logic of Separating Modalities. Manuscript, UCL, , 2015.
13. D. Galmiche, D. Méry, and D. Pym. The Semantics of BI and Resource Tableaux. *Mathematical Structures in Computer Science*, 15:1033–1088, 2015.
14. T. Hoare. Generic Models of the Laws of Programming. *LNCS*, 8051:213–226, 2013.
15. J.-B. Jeannin, D. Kozen, and A. Silva. Language Constructs for Non-well-Founded Computation. In *Proc. 22nd ESOP*, pages 61–80. Springer-Verlag Berlin, Heidelberg, 2013.
16. Hewlett-Packard Laboratories. Towards a science of risk analysis. `http://www.hpl.hp.com/news/2011/oct-dec/security_analytics.html`. Accessed 16 October 2015.
17. Dominique Larchey-Wendling and Didier Galmiche. Exploring the relation between Intuitionistic BI and Boolean BI: an unexpected embedding. *Mathematical Structures in Computer Science*, 19(3):435–500, 2009.
18. S. Milius. A sound and complete calculus for finite stream circuits. In *Proc. 25th LICS*, pages 421–430, 2010.
19. P. O'Hearn. Resources, concurrency, and local reasoning. *Theoretical Computer Science*, 375(1–3):271–307, 2007.
20. P. O'Hearn and D. Pym. The logic of bunched implications. *Bulletin of Symbolic Logic*, 5(2):215–244, June 1999.
21. D. Pym, P. O'Hearn, and H. Yang. Possible Worlds and Resources: The Semantics of BI. *Theoretical Computer Science*, 315(1):257–305, 2003.
22. S. Read. *Relevant Logic*. Basil Blackwell, 1988.
23. J. Reynolds. Separation logic: a logic for shared mutable data structures. In *Proc. of 17th LICS, IEEE*, 2002.
24. Y. Shoham and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2008.