When Are Two Workflows the Same?

Jan Hidders[1], Marlon Dumas[2], Wil M.P. van der Aalst[3], Arthur H.M. ter Hofstede[2], Jan Verelst[4]

[1] Dept. of Mathematics and Computer Science, University of Antwerp, Antwerp, Belgium
E-Mail: jan.hidders@ua.ac.be
[2] Centre for IT Innovation, Queensland University of Technology, Brisbane, Australia
E-Mail: {m.dumas,a.terhofstede}@qut.edu.au
[3] Dept. of Technology Management, Eindhoven University of Technology, Eindhoven, The Netherlands
E-Mail: W.M.P.v.d.Aalst@tm.tue.nl
[4] Dept. of Management Information Systems, University of Antwerp, Antwerp, Belgium
E-Mail: jan.verelst@ua.ac.be

Abstract

In the area of workflow management, one is confronted with a large number of competing languages and the relations between them (e.g. relative expressiveness) are usually not clear. Moreover, even within the same language it is generally possible to express the same workflow in different ways, a feature known as variability. This paper aims at providing some of the formal groundwork for studying relative expressiveness and variability by defining notions of equivalence capturing different views on how workflow systems operate. Firstly, a notion of observational equivalence in the absence of silent steps is defined and related to classical bisimulation. Secondly, a number of equivalence notions in the presence of silent steps are defined. A distinction is made between the case where silent steps are visible (but not controllable) by the environment and the case where silent steps are not visible, i.e., there is an alternation between system events and environment interactions. It is shown that these notions of equivalence are different and do not coincide with classical notions of bisimulation with silent steps (e.g. weak and branching).

1 Introduction

Workflow systems support the coordination of manual and automated activities on the basis of explicit process models. In the last two decades, there has been significant interest in the possibilities offered by these systems and other similar tools for automating business processes. Unfortunately, a lack of formal foundation coupled with failed standardization efforts have led to a plethora of similar but subtly different workflow modeling languages.

The resulting babel has raised the issue of comparing the relative expressiveness between languages and translating models defined in one language into "equivalent" models defined in another language. In addition, even within the same language it is often possible to define multiple "equivalent" models of the same workflow. This in turn raises the following questions: (1) In which ways are these workflow models different? (2) Are these differences significant or is their nature superficial and of limited consequence? (3) If they are significant, how can they be dealt with?

We use the term *variability* to refer to the possibility of defining multiple models of a workflow in a given modeling language. We are not aware of any research that systematically categorizes variability in workflow modeling. In the field of conceptual modeling of information systems, variability is a recognized phenomenon. For example, (Verelst 2004) proposes a framework in which three types of variability are identified in UML class diagrams and Entity Relationship diagrams. In this framework, variability in conceptual modeling was shown to be of a rather fundamental nature: even under considerable restrictions (differences in layout and notation were not considered; only differences between models in the same modeling language), it was shown to be possible to model the same set of concepts in the real world in very different ways. Following this experience in the conceptual modeling field, we have observed that also in the workflow field there are examples of variability, thus raising a number of issues. For instance, two modelers may produce different models when faced with the same modeling problem. This in turn may hinder the reuse of workflow models (Janssens, Verelst & Weyn 1998), since it makes it difficult to compare a (sketch of) workflow model required by a given modeler with workflow models available for reuse. A similar problem arises in the setting of inter-organizational workflows, where it is sometimes necessary to compare a model of a workflow required by a partner with that provided by another (Wombacher & Mahleko 2002).

In order to study the relative expressiveness of workflow languages and to deal with variability in workflow modeling, a formal foundation is needed that defines what "equivalence of workflow models" means. The work reported in this paper aims at contributing to the establishment of this formal foundation. The specific question addressed can be stated as follows: When and why two workflows can be considered to be equivalent? The paper provides elements of an answer to this question while abstracting from the language used to describe workflows. The results can be applied to specific languages or pairs thereof, like for example WF nets (van der Aalst 1998), YAWL (van der Aalst & ter Hofstede 2003), or BPEL (Andrews, Curbera, Dholakia, Goland, Klein, Leymann, Liu, Roller, Smith, Thatte, Trickovic & Weerawarana 2003), but this is outside the scope of this paper.

Workflows (sometimes called "workflow processes" or "business processes" although the latter term is used with various connotations) can be seen from a number of perspectives (Jablonski & Bussler 1996). The *control-flow* (or process) perspective describes the execution ordering between the basic activities involved in the workflow. The *data* (or information)

perspective captures the structure of the data involved in the execution of activities and how data are passed between activities. The *resource* (or organizational) perspective provides an organizational anchor to the workflow and determines the resources that are involved in the execution of activities. The *operational* perspective describes the internal structure of activities by mapping them to manual or automated actions. In terms of the languages used to specify workflows, the control flow perspective plays a central role. Indeed, the data perspective is constrained by it (and can be defined on top of it), while the organizational and operational perspectives are ancillary. Hence, this paper considers workflows abstracted at the level of control flow.

In previous work, well-known notions of equivalence such as trace equivalence (Wombacher & Mahleko 2002), weak bisimulation (Kiepuszewski, ter Hofstede & van der Aalst 2003), and branching bisimulation (van der Aalst & Basten 2002) have been used in the area of workflow.[1] Practical examples show that trace equivalence is too weak. Accordingly, most authors have adopted either weak or branching bisimulation. However, there has been no formal argumentation as to which notion of equivalence is more appropriate for workflow modeling. We argue that a well-motivated notion of equivalence for workflows is imperative to study expressiveness and for defining (possibly partial) inter-language process mappings.

Indeed, when defining mappings between different languages, one is often confronted with the situation where a language defines a construct that can be mapped in several ways in terms of constructs of the other language. For example, the "OR-split" construct represents a multi-choice between a number of branches, such that none, some or all branches are chosen, and if several branches are chosen they are executed in parallel. This construct is not supported by all workflow languages (van der Aalst, ter Hofstede, Kiepuszewski & Barros 2003) but it can be mapped into a combination of constructs for parallel execution and exclusive choice. Specifically, an OR-split leading to two tasks A and B can be translated into either an exclusive system-controlled choice between doing nothing, only A, only B, or both A and B in parallel. An alternative translation is a parallel execution of two branches: one in which an exclusive system choice is made between A or nothing, and another branch where an exclusive system choice is made between B or nothing. To be able to properly capture the OR-split pattern in a given language, it is imperative to have a well-defined notion of equivalence that captures the language's semantics.

In this setting, this paper defines notions of equivalence from an observational viewpoint that reflect the way(s) workflow systems operate. Observational equivalence is defined in terms of the sets of tasks that the workflow offers to its environment in response to the inputs that the environment provides. Such sets of tasks are known as *work-sets* (or "worklists" when they are prioritized) and are central to generally accepted conceptions of workflow systems (Hollingsworth 1995).

Where possible, the observational equivalence notions defined are related to known bisimilarity relations. In particular, we define six notions of equivalence for workflows with silent steps, and we show that these notions are distinct and that none of them coincides with either weak or branching bisimulation.

[1] Note that concurrent bisimulation and related notions have not been considered in workflow systems. Instead, each task in a workflow is expanded into a "begin task" and an "end task" and equivalence notions are then defined on the expanded workflow (a typical approach in process algebra (Baeten & Weijland 1990)). This approach is followed in e.g. (Kiepuszewski et al. 2003).

The paper is structured as follows. Section 2 considers the case of workflow models without silent steps (but with non-determinism). A notion of observational equivalence is defined under this assumption and related to bisimulation. Section 3 considers the case where silent steps are allowed. Notions of observational equivalence for this general case are defined and classified according to whether they make silent steps visible to the environment or not. Finally, Section 4 concludes.

2 Workflows without Silent Steps

We begin this section with an informal description of what we define as a workflow and its observable behavior. Based upon this description we then present a formalization of these notions.

A *workflow* is a set of activities related by control-flow dependencies. When executed by a *workflow (management) system*, a workflow behaves as a reactive system where alternatively the system makes an offer to the environment in the form of a set of activity identifiers, and the environment responds with a choice of one of the elements in the offer and any additional information that is required to complete the activity.

To remain at a high level of abstraction, we describe the behavior of a workflow with a *transition tree*, i.e., a connected, rooted, edge-labeled and directed graph without cycles. For example, a workflow that first offers the set $\{A, B\}$ and then always offers $\{C\}$ is represented as shown in Figure 1. This

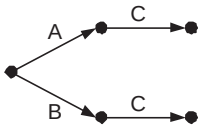

Figure 1: A transition tree

definition is independent of the language used to describe workflows. Typically, such languages rely on constructs such as task node, parallel split nodes, synchronization nodes, decision/choice nodes, merge nodes, etc. But in any case, the resulting descriptions can be "expanded" into transition trees.

The above definition of workflows suggests that the observable behavior of a workflow can be described as a possibly infinite set of traces that consist of alternations of offers, containing a set of activity identifiers, and acceptances, containing a single activity identifier that is an element of the preceding offer. Since this describes the observable behavior of the workflow, two workflows should then be considered the same if they define the same set of traces. Note that, up to this point, we could adopt the usual definition of traces by not including the offers in the trace (i.e., we could define a trace as a list of activity identifiers) and this would lead to the same equivalence relation.

However, as is often argued in concurrency theory, such a trace-based model does not seem to capture all there is to know about the control flow since the "moment of choice" is not represented. The standard example is the distinction between the transition trees in Figure 2. In T_1 it is modeled that the choice of how the process will end is taken at the step A whereas in T_2 it is taken at the step B. From a workflow perspective, this distinction captures the fact that the environment does not just pick an activity identifier (say A), but in addition to this, it supplies external information needed to conclude the activity. For example, in the case of an activity "Get age", when the

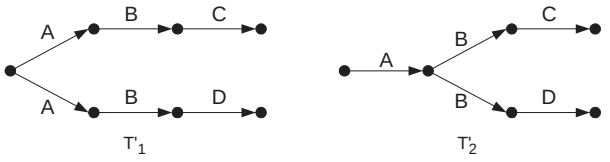

Figure 2: Abstract transition trees T_1 and T_2

environment picks this activity it must supply the requested information. This information may then be used by the system that executes the workflow to determine the offer to be made in the next step (e.g., different continuations may be observed depending on the age entered). Therefore the "moment of choice" is observable by the environment because it can see whether certain information that was supplied made a difference for the subsequent offers that were made. Indeed, if the logs of a workflow would contain the supplied information we would be able to tell the difference between a workflow described by T_1 and one described by T_2.

Consider, for example, the two transition trees in Figure 3 where we label the edges not only with the activity identifiers (in uppercase) but also with the supplied information (in lowercase). In this example, the supplied information correspond to decisions ('y' for *yes*, and 'n' for *no*) by the environment. It should

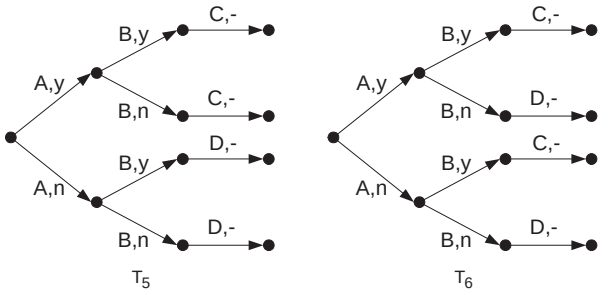

Figure 3: Concrete transition trees T_1' and T_2'

be clear that T_1 in Figure 2 is an abstraction of T_1' but not of T_2', and T_2 is an abstraction of T_2' but not of T_1'. Moreover, these two concrete transition trees can be distinguished by their traces and therefore by their observable behavior, because the environment can see which of the decisions actually matter for the final offer. Thus the intuition of "moment of choice" seems to be captured by the assumption that the (abstract) transition tree is an abstraction of the concrete transition tree that ignores the information that is provided by the environment.

In process algebra the intuition about "moment of choice" is captured by the notion of bisimulation. In the following we will give a formal argument that shows that the intuition about externally supplied information indeed justifies the use of this equivalence notion for workflows.

We begin by postulating an infinite set $\mathcal{A}$ of *activity identifiers* such as "Receive goods", "Check credit", "Get patient information". These will be the activities that are offered to the environment (i.e., set of users and applications that interact with the workflow system). Next to this set we assume an infinite set $\mathcal{I}$ of *input data*. These represent units of information that can be supplied by the environment of the system for the completion of an activity. Examples of such data are a simple boolean such as for the activity "Check credit" or a complex data structure such as for "Get patient information". We also postulate an infinite set $\mathcal{V}$ of nodes.

Definition 2.1 (Abstract Workflow). An *abstract workflow* (AW) is a tuple (V, E, r) which represents a rooted edge-labeled tree with nodes $V \subseteq \mathcal{V}$, edges $E \subseteq V \times \mathcal{A} \times V$, and root $r \in V$.

As already explained an abstract workflow abstracts from the data that is supplied by the external environment. Therefore we introduce the notion of concrete workflow that does take this information into account by labeling the edges with a pair (a, i) where a is an activity identifier and i the externally supplied information.

Definition 2.2 (Concrete Workflow). A *concrete workflow* (or CW) is a tuple (V, E, r) which represents a rooted edge-labeled tree with nodes $V \subseteq \mathcal{V}$, edges $E \subseteq V \times (\mathcal{A} \times \mathcal{I}) \times V$, and root $r \in V$.

We will require concrete workflow to be consistent, which means that a certain activity always accepts the same set of input data. More formally we call a CW (V, E, r) *consistent* if it holds that if there is an edge $(n_1, (a, i), n_2) \in E$ and an edge $(n_3, (a, j), n_4) \in E$ then there is also an edge $(n_1, (a, j), n_5) \in E$. Finally we introduce the usual notion of determinism for CWs such that every specific choice and input action by the environment leads always to the same state. In other words, a system that executes a deterministic process does not make arbitrary choices. Formally, a CW (V, E, r) is called *deterministic* if for every node $n \in V$ and pair $(a, i) \in \mathcal{A} \times \mathcal{I}$ there is at most one edge $(n, (a, i), n') \in E$. Note that in deterministic CWs the environment can still make arbitrary choices when selecting an activity or supplying input information.

The interpretation of a CW is that in each state the system offers the environment a choice in the form of a work-set: a set of identifiers corresponding to the activities that the environment can perform in this state. The environment then makes a choice a from the work-set and supplies the information i, after which the system moves to the state that is indicated by the edge labeled with (a, i). The work-set is more formally defined as follows. Given a CW (V, E, r) the *work-set* of a node $n \in V$, denoted as $W(n)$, is defined such that $W(n) = \{a | (n, (a, i), n') \in E\}$.

Because we want to base our notion of equivalence on observational equivalence we have to define what it exactly is that is observed about the workflow system. For this purpose we introduce the notion of workflow trace. The events we record in this trace are (1) the work-set that the system offers to the environment and which consists of a set of activity identifers; (2) the choice that the environment makes from this work-set, represented by an activity identifer; and (3) the data that is supplied by the environment for completing the activity, represented by an element of $\mathcal{I}$. This leads to the following definition.

Definition 2.3 (Workflow Trace). The *set of workflow traces* of a CW (V, E, r) is defined as a set of lists of the form $(2^{\mathcal{A}} \cdot \mathcal{A} \cdot \mathcal{I})^* \cdot 2^{\mathcal{A}}$ such that the list $\langle W_1, a_1, i_1, \ldots, W_k, a_k, i_k, W_{k+1} \rangle$ is in this set iff there is a path $\langle (n_1, (a_1, i_1), n_2), \ldots, (n_k, (a_k, i_k), n_{k+1}) \rangle$ in E such that $r = n_1$ and $W_i = W(n_i)$ for each $1 \leq i \leq k + 1$.

The final extra offer W_{k+1} in a trace is necessary to distinguish the trees shown in Figure 4.

We note that this definition of workflow trace is equivalent to that of colored trace defined in (van Glabbeek & Weijland 1996) (also called *decorated trace* by some authors) if we equate an offer to a color.

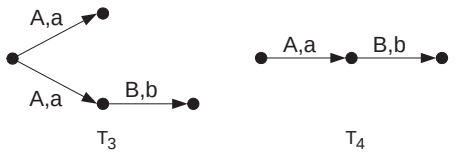

Figure 4: The final offer is needed to be able to distinguish transition trees T_3 and T_4

It is easy to see that deterministic CWs are characterized up to isomorphism by their workflow traces, i.e., two *deterministic* CWs are isomorphic iff they have the same set of workflow traces (Engelfriet 1985).

We proceed with establishing the relationship between CWs and AWs. This is done with an instance relation that relates the nodes in the AW and the CW. We will require that such a relation at least relates the root nodes. Furthermore, for each abstract transition in the AW there should at least be one corresponding concrete transition in the CW that justifies its existence. On the other hand, every concrete transition in the CW must be justified by a corresponding abstract transition in the AW.

Definition 2.4 (Instance Relation). An *instance relation* between an AW (V_1, E_1, r_1) and CW (V_2, E_2, r_2) is a relation $H \subseteq V_1 \times V_2$ such that (1) $H(r_1, r_2)$, (2) if $(n_1, a, n_1') \in E_1$ and $H(n_1, n_2)$ then there is an edge $(n_2, (a, i), n_2') \in E_2$ such that $H(n_1', n_2')$, and (3) if $(n_2, (a, i), n_2') \in E_2$ and $H(n_1, n_2)$ then there is an edge $(n_1, a, n_1') \in E_1$ and $H(n_1', n_2')$.

Using this notion we can now define what a concrete instance of an abstract workflow is.

Definition 2.5 (Instance). A CW T is called an *instance* of an AW T' if (1) T is consistent and (2) there is an instance relation between T' and T.

The similarity between the instance relation and a bisimulation relation leads to the following observation.

Proposition 2.1. *A consistent CW T is an instance of an AW T' iff the AW T'' that is obtained from T by replacing all edge labels (a, i) with a is bisimilar with T'.*

Proof. (Sketch) It can be shown that the instance relation H from T' to T indeed defines a bisimulation relation between T'' and T'. Vice versa it can be shown that the bisimulation relation B from T'' to T' indeed defines an instance relation from T to T'. □

Since we already had a notion of observational equivalence for CWs in the form of sets of workflow traces, we can now extend this notion to AWs.

Definition 2.6 (Observational Equivalence). Two AWs are called *observation equivalent* if the sets of workflow trace sets of their instances are the same.

Note that this indeed captures the intuition that two AWs are distinguishable iff it is possible, by looking at the workflow trace sets, to detect an instance that belongs to the one but not to the other. The definition of an instance relation is very similar to that of a bisimulation relation, which leads to the following lemma.

Lemma 2.2. *Two AWs are bisimilar iff they have the same set of instances.*

Proof. (Sketch) The only-if part is shown by demonstrating that a consistent CW is an instance iff it is bisimilar to the AW if we ignore the input data, and the fact that the bisimulation relation is an equivalence relation. The same fact can be used to show that the if-part follows because every AWS has at least one instance and so the two AWs will have at least one common instance and therefore be bisimilar. □

It is not hard to see that for every AW there is a deterministic instance. Since we already observed that two deterministic CWs have the same trace set iff they are isomorphic, it follows that if we had only allowed deterministic instances then we would have indeed by now have shown that two AWs are bisimilar iff they are observation equivalent.

In order to make the proposed equivalence notions as general as possible, we now consider the case of non-deterministic CWs. The problem with non-deterministic CWs is that different edges with identical labels may leave from the same node and therefore the CWs are no longer defined up to isomorphism by the set of workflow traces. Consider the examples in Figure 5.

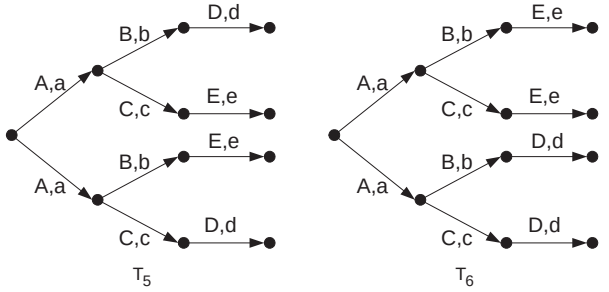

Figure 5: Non-deterministic CWs

Both CWs have the same trace sets:

1. $\langle\{A\}, A, a, \{B, C\}, B, b, \{D\}, D, d, \emptyset\rangle$
2. $\langle\{A\}, A, a, \{B, C\}, C, c, \{E\}, E, e, \emptyset\rangle$
3. $\langle\{A\}, A, a, \{B, C\}, B, b, \{E\}, E, e, \emptyset\rangle$
4. $\langle\{A\}, A, a, \{B, C\}, C, c, \{D\}, E, d, \emptyset\rangle$

This is not surprising since our type of trace-equivalence for CWs is essentially the same as *ready trace semantics* which is known to be less strict than bisimulation but more strict than the standard trace semantics (van Glabbeek 1993). However, this is at the level of CWs and in the following it will be shown that it is not a problem at the level of AWs.

We already observed that deterministic CWs are characterized up to isomorphism by their trace sets. We will now show a similar result for a wider class, viz., the CWs that are deterministic up to bisimulation.

Definition 2.7 (Deterministic up to bisimulation). A CW $T = (V, E, r)$ is said to be *deterministic up to bisimulation* if for every two distinct edges $(n_1, (a, i), n_2)$ and $(n_1, (a, i), n_3)$ in E it holds that n_2 and n_3 are bisimilar in T.

An alternative characterization of this weaker form of determinism is given by the following proposition.

Proposition 2.3. *A CW is deterministic up to bisimulation iff it is bisimilar to a deterministic CW.*

Proof. (Sketch) The if-part holds because if $(n_1,(a,i),n_2)$ and $(n_1,(a,i),n_3)$ in the CW then n_2 and n_3 will be bisimilar to the same node in the deterministic CW, and therefore also bisimilar themselves.

The only-if part holds because we can simply merge any two edges $(n_1,(a,i),n_2)$ and $(n_1,(a,i),n_3)$ with n_2 and n_3 bisimilar in the CW, until no more such edges are found (possibly only after an infinite number of steps). The result will be bisimilar to the original and deterministic. □

Although such CWs are not characterized by their workflow traces up to isomorphism, it can be shown that they are characterized by their workflow traces up to bisimulation.

Lemma 2.4. *If two CWs are deterministic up to bisimulation then they have the same set of workflow traces iff they are bisimilar.*

Proof. (Sketch) We first prove the if part. For every path $\langle(n_1,(a_1,i_1),n_2),\ldots,(n_{k-1},(a_k,i_k),n_k)\rangle$ from the root in one CW there will be a similar path $\langle(n'_1,(a_1,i_1),n'_2),\ldots,(n'_k,(a_k,i_k),n'_{k+1})\rangle$ from the root in the other CW such that for all $1 \leq i \leq k+1$ the nodes n_i and n'_i are bisimilar, and therefore define the same work-set W_i.

Now we prove the only-if part. Let T_1 and T_2 be the two CWs that are deterministic up to bisimulation. A CW that is deterministic up to bisimulation is bisimilar with a deterministic CW, viz., the one we obtain when we merge the edges $(n_1,(a,i),n_2)$ and $(n_1,(a,i),n_3)$ for which n_2 and n_3 are bisimilar. Because two CWs have the same set of workflow traces if they are bisimilar this deterministic CW has the same set of workflow traces. Since deterministic CWs are characterized up to isomorphism by their workflow traces, it follows that for both T_1 and T_2 the corresponding deterministic CWs are isomorphic, and therefore that T_1 and T_2 are bisimilar. □

Another useful observation is that the CWs that are deterministic up to bisimulation and those that are not, are observationally distinct, i.e., they cannot have the same set of workflow traces.

Lemma 2.5. *If T_1 is a CW that is deterministic up to bisimulation, and T_2 a consistent CW that is not, then T_1 and T_2 cannot have the same set of workflow traces.*

Proof. (Sketch) Suppose we try to make T_2 deterministic by merging pairs of edges of the form $(n_1,(a,i),n_2)$ and $(n_1,(a,i),n_3)$ until we find no more such pairs of edges (possibly after an infinite number of steps). If during this process at each step we only merged nodes that had the same abstract work-set (and because T_2 is consistent therefore also the same concrete work-set) then we know that (1) the result is bisimilar to T_2 and (2) the result has the same set of workflow traces as T_2. However, by Proposition 2.3 it would then follow that T_2 is deterministic up to bisimulation, which contradicts the original assumption. Therefore there must have been a step were we merged nodes with different abstract work-sets. The paths to these nodes must therefore define two different workflow traces that are the same except for the last work-set. Since the merging does not change the set of workflow traces it follows that these are workflow traces of T_2. It is clear that such workflow traces can never be in the set of workflow traces of a deterministic CW. Moreover, by Proposition 2.3 and the fact that two CWs have the same set of workflow traces if they are bisimilar we know that the set of workflow traces of T_1 must be equal to that of a deterministic CW. So it follows that these traces cannot be in the set of workflow traces of T_1. □

Finally we are ready to show the main theorem of this section that states that for AWs the notions of bisimulation and observation equivalence coincide.

Theorem 2.6. *Two AWs are bisimilar iff they are observation equivalent.*

Proof. (Sketch) We first show the if-part. Assume that the AWs T_1 and T_2 are not bisimilar. Then we construct the CW T'_1 by taking T_1 and replacing the activity identifiers a with a pair (a,i) where $i \in I$ such that we obtain a deterministic CW. Note that this is possible because we assume that $\mathcal{I}$ is infinite. Since T_1 and T_2 are not bisimilar it follows that there is no instance of T_2 that is bisimilar to T'_1. So let us assume that there is an instance of T_2 that is not bisimilar to T'_1 but still has the same set of workflow traces. This CW cannot be deterministic up to bisimulation because by Lemma 2.4 it would then be bisimilar to T'_1. However, if it is not deterministic up to bisimulation then it follows by Lemma 2.5 that it could not have the same set of workflow traces as T'_1. It follows then there cannot be an instance of T_2 that has the same set of workflow traces as T'_1. Consequently there is a set of workflow traces for T_1 that is not a set of workflow traces from T_2 and therefore the two are not observation equivalent.

We now proceed with the only-if-part. By Lemma 2.2 we know that two bisimilar AWs have the same instances, and therefore they also have the same sets of workflow traces. □

3 Workflows with Silent Steps

In the area of concurrency theory, it is common to distinguish between steps of a process that are directly observable and controllable by the environment, and those that are not. An observable step occurs if the process is ready to perform it (in workflow terms: the corresponding task appears in the offered work-set) and the environment allows it to occur (e.g., the task is picked from the work-set). In contrast, the silent step (denoted τ) is entirely controlled by the system and the environment cannot directly observe its occurrence. In Workflow nets (van der Aalst 1998) for example, τ steps appear in the form of unlabeled transitions and play an important role in describing the semantics of `xor-split`, `and-split`, `or-join` and `and-join` nodes in workflow models. They correspond to internal tasks and decisions performed by the workflow system, such as for example updating a variable of the workflow, evaluating a boolean condition to select an execution path, executing a script within the workflow system, or synchronizing tasks. These types of steps are system-controlled (i.e., the system decides when and how to do them), as opposed to the other steps (e.g., performing a human or automated task) that are interpreted as environment-controlled steps. This, however, does not indicate to what extent they should be considered part of the observable behavior of the workflow. Indeed, even if the environment cannot directly observe the occurrence of a τ step, it can indirectly detect it if the process has less options for continuation after the occurrence than before (i.e., if it causes tasks to be added or removed from the offered work-set).

It turns out that the introduction of silent steps has strong implications in terms of workflow equivalence. To discuss these implications, we adapt the definitions given in Section 2 to workflows with silent steps as follows:

- The definition of AW remains unchanged except that a distinguished task $\tau \in \mathcal{A}$ is introduced.
- The definition of CW remains unchanged except that a constraint is added to enforce that the activity identifier τ is only associated to the empty input data (denoted $\perp$), i.e., if an edge is labeled with (τ, i) then $i = \perp$.
- The definitions of instance relation and instance remain unchanged.
- The definition of work-set $W(n)$ will exclude τ, i.e., for a CW (V, E, t) we define $W(n) = \{a | (n, (a, i), n') \in E \wedge a \neq \tau\}$.
- The definitions of workflow trace and observational equivalence vary depending on the interpretation of silent steps as discussed below.

3.1 Classical notions of equivalence

In this section we present two existing notions of equivalence for processes with τ steps: weak bisimulation (Milner 1980) and branching bisimulation (Basten 1996, van Glabbeek 1993).

In the following of this paper we use $s \Rightarrow s'$ to denote a path of zero or more τ edge from s to s'.

Definition 3.1 (Weak Bisimilarity). Two AWs T_1 and T_2 are *weakly bisimilar*, denoted $\equiv_{wbs}$, iff there exists a symmetric relation R between the nodes of T_1 and T_2 such that:

1. The roots are related by R
2. If $R(r, s)$ and $r \stackrel{a}{\Rightarrow} r'$ then there exists a path $s \stackrel{a}{\Rightarrow} s'$ such that $R(r', s')$, where $r \stackrel{a}{\Rightarrow} r'$ denotes a path $r \Rightarrow r_1 \stackrel{a}{\rightarrow} r_2 \Rightarrow r'$ if $a \neq \tau$ and a path $r \Rightarrow r'$ if $a = \tau$.

If we consider the AWs that correspond to the CWs in Figure 6 then it can be seen that $T_7 \equiv_{wbs} T_9$, T_{10}, and T_{12}. On the other hand AW T_8 is not equivalent to any of the other corresponding AWs. The same applies to T_{11}.

The intuition behind this notion of equivalence is that every activity including any preceding and following τ steps should be mirrored by a similar path in the "equivalent workflow".

Definition 3.2 (Branching Bisimilarity). Two AWs T_1 and T_2 are *branching bisimilar*, denoted $\equiv_{bbs}$, iff there exists a symmetric relation R between the nodes of T_1 and T_2 such that:

1. The roots are related by R
2. If $R(r, s)$ and $r \stackrel{a}{\rightarrow} r'$ then either $a = \tau$ and $R(r', s)$ or there exists a path $s \Rightarrow s_1 \stackrel{a}{\rightarrow} s'$ such that $R(r, s_1)$ and $R(r', s')$.

For the AWs in Figure 6 we get the same equivalence classes for $\equiv_{bbs}$ as for $\equiv_{wbs}$: $\{T_7, T_9, T_{10}, T_{12}\}$, $\{T_8\}$ and $\{T_{11}\}$.

The intuition behind these two equivalence notions is that in order to "mirror" a step A of a given process, it is possible to precede and follow A with a number of τ steps in the "equivalent" process. The difference between the two notions is that branching bisimilarity preserves the branching structure of the process by further imposing that all the τ steps taken before A lead to states that offer identical sets of possible future choices, and similarly all the τ steps taken after A must lead to states that offer identical sets of possible future choices (though not necessarily the same set of future choices as before A). A comparison of branching bisimilarity, weak bisimilarity, and other equivalences can be found in (van Glabbeek & Weijland 1996).

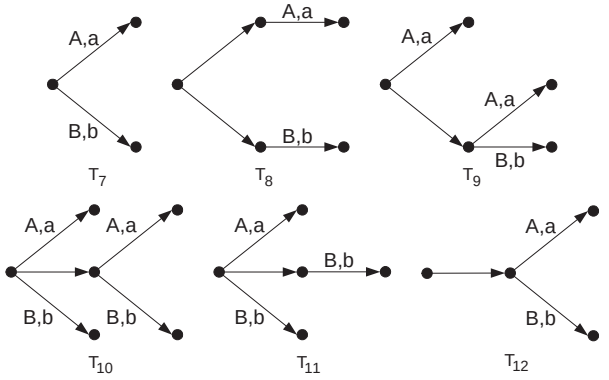

Figure 6: Examples illustrating the concept of silent steps

Both notions of equivalence are generalization of bisimulation and seem good candidates for defining the semantics of workflows. We will investigate this in the following sections by comparing them to several observational semantics that seem reasonable and correspond to different interpretations of the τ steps. To structure the presentation we consider two options in turn: (i) we let τ steps appear in the traces but then define an equivalence relationship over the traces that captures when certain τ steps cannot be observed; and (ii) traces are defined such that τ steps do not appear in them.

3.2 Semantics with visible silent steps

We explore three types of semantics where τ steps may be observed in the traces: *full semantics*, *change semantics*, and *non-empty semantics*.

3.2.1 Full semantics

The full semantics considers each τ step to be visible. This semantics can be formalized simply by taking Definition 2.3 but with the new notion of work-set, i.e., we exclude τ from the offered work-sets W_i. For example, T_7 has two possible traces: $\langle \{A, B\}, A, a, \emptyset \rangle$ and $\langle \{A, B\}, B, b, \emptyset \rangle$. T_8 also has two possible traces: $\langle \emptyset, \tau, \perp, \{A\}, A, a, \emptyset \rangle$ and $\langle \emptyset, \tau, \perp, \{B\}, B, b, \emptyset \rangle$. With this modified version of Definition 2.3, the definition of observational equivalence under full semantics for AWs, denoted $\equiv_{fs}$, is the same as Definition 2.6. Clearly, all six AWs in Figure 6 can be distinguished using this semantics.

3.2.2 Change semantics

The change semantics considers only τ steps that result in a modified work-set to be visible. The interpretation of this semantics is that the environment cannot see explicit τ steps but it can see changes in the activities offered by the system. The traces of T_7, T_8, T_9, T_{11}, T_{12} are identical to the full semantics. However, for T_{10} there is a difference between the full semantics and the change semantics. In the change semantics there are two possible traces ($\langle \{A, B\}, A, a, \emptyset \rangle$ and $\langle \{A, B\}, B, b, \emptyset \rangle$) while in the full semantics there are four possible traces: $\langle \{A, B\}, A, a, \emptyset \rangle$, $\langle \{A, B\}, B, b, \emptyset \rangle$, $\langle \{A, B\}, \tau, \perp, \{A, B\}, A, a, \emptyset \rangle$, and $\langle \{A, B\}, \tau, \perp, \{A, B\}, B, b, \emptyset \rangle$. We use $\equiv_{cs}$ to denote

that two AWs are observation equivalent under the change semantics. This can be formalized as follows.

Definition 3.3 (Equivalence relation of traces under change semantics). Let $\equiv^t_{cs}$ be the smallest equivalence relation such that for any pair of workflow traces $t_1 = \langle X_1, A_1, a_1, \ldots, X_n, \tau, \bot, X_{n+1}, A_{n+1}, a_{n+1}, X_{n+2}, \ldots, X_m\rangle$ and $t_2 = \langle X_1, A_1, a_1, \ldots, X_n, A_{n+1}, a_{n+1}, X_{n+2}, \ldots, X_m\rangle$: $t_1 \equiv^t_{cs} t_2$ if $X_n = X_{n+1}$.

Informally the requirement for $\equiv^t_{cs}$ can be formulated as follows: if a trace t contains a sublist of the form $X, \tau, \bot, X$ and we replace this sublist in t with X then we obtain an equivalent trace.

Definition 3.4 (Equivalence relation of CWs and AWs under change semantics). Let T_1 and T_2 be two CWs, $T_1 \equiv_{cs} T_2$ iff in the full semantics for every workflow trace t_1 of T_1 there exists a workflow trace t_2 of T_2 such that $t_1 \equiv^t_{cs} t_2$ and vice versa.

Let T_1 and T_2 be two AWs, then $T_1 \equiv_{cs} T_2$ iff for every instance T_1' of T_1 there is an instance T_2' of T_2 such that $T_1' \equiv_{cs} T_2'$ and vice versa.

It follows that for any two AWs T_1 and T_2, $T_1 \equiv_{fs} T_2$ implies $T_1 \equiv_{cs} T_2$ but the implication does not necessarily hold the other way, i.e., $\equiv_{fs} \subset \equiv_{cs}$. The inclusion is strict since in Figure 6 $T_{10} \not\equiv_{fs} T_7$ but $T_{10} \equiv_{cs} T_7$.

3.2.3 Non-empty semantics

The non-empty semantics abstracts from τ steps that occur when the work-set is empty. The interpretation of this semantics is that the environment cannot see explicit τ steps that leave from a state where no actions are offered to the environment. If we remove from the traces the empty offers that are followed by a τ step then the traces of T_7, T_9, T_{10}, and T_{11} remain identical. However, for T_8 and T_{12} we obtain different traces; for T_8 we get $\langle\{A\}, A, a, \emptyset\rangle$ and $\langle\{B\}, B, b, \emptyset\rangle$, and for T_{12} we get $\langle\{A, B\}, A, a, \emptyset\rangle$ and $\langle\{A, B\}, B, b, \emptyset\rangle$. We write $\equiv_{nes}$ to denote that two AWs are observation equivalent under the non-empty semantics.[2] This can be formalized as follows.

Definition 3.5 (Equivalence relation of traces under non-empty semantics). Let $\equiv^t_{nes}$ be the smallest equivalence relation such that for any pair of workflow traces $t_1 = \langle X_1, A_1, a_1, \ldots, X_n, A_n, a_n, \emptyset, \tau, \bot, X_{n+2}, \ldots, X_m\rangle$ and $t_2 = \langle X_1, A_1, a_1, \ldots, X_n, A_n, a_n, X_{n+2}, \ldots, X_m\rangle$: $t_1 \equiv^t_{nes} t_2$.

Informally the requirement for $\equiv^t_{nes}$ can be formulated as follows: If a trace t contains a sublist of the form $\langle\emptyset, \tau, \bot\rangle$ and we remove this sublist from t then we obtain an equivalent trace.

Definition 3.6 (Equivalence relation of CWs and AWs under non-empty semantics). Let T_1 and T_2 be two CWs, $T_1 \equiv_{nes} T_2$ iff in the full semantics for every workflow trace t_1 of T_1 there exists a workflow trace t_2 of T_2 such that $t_1 \equiv^t_{nes} t_2$ and vice versa.

Let T_1 and T_2 be two AWs, then $T_1 \equiv_{nes} T_2$ iff for every instance T_1' of T_1 there is an instance T_2' of T_2 such that $T_1' \equiv_{nes} T_2'$ and vice versa.

It follows that for any two AWs T_1 and T_2, $T_1 \equiv_{fs} T_2$ implies $T_1 \equiv_{nes} T_2$ but the implication does not necessarily hold in the other way, i.e., $\equiv_{fs} \subseteq \equiv_{nes}$. Also this inclusion is strict since in Figure 6 $T_7 \not\equiv_{fs} T_{12}$ but $T_7 \equiv_{nes} T_{12}$. Moreover, there are AWs T and T' such that $T \equiv_{nes} T'$ but $T \not\equiv_{cs} T'$, e.g., T_7 and T_{12} in Figure 6. Similarly, there may be AWs T and T' such that $T \equiv_{cs} T'$ but $T \not\equiv_{nes} T'$, e.g., T_7 and T_{10}.

Surprisingly none of the above three semantical definitions coincides with standard equivalence notions such as branching bisimulation or weak bisimulation. Of the three equivalence notions given, the change semantics seems to be the most suitable one. Indeed, it corresponds to the case where the workflow system updates the work-set immediately after every step (whether the step is internal to the system or the result of an external action) and makes the new work-set visible to the environment. Hence, the environment is able to detect the occurrence of any τ step that changes the contents of the work-set.

Unfortunately, an elegant bisimulation-like formulation of these notions seems unlikely, as evidenced by the two AWs in Figure 7. Here $T_1 \equiv_{cs} T_2$ and so the bisimulation relation would have to relate the root nodes of T_1 and T_2 but at the same time relate the end node of the first τ edge in T_1 with the end node of the second τ edge in T_2 and vice versa. The reason for this is that in the visible step semantics, the environment can observe the options are available to it (i.e., the work-set) without having to try an option.

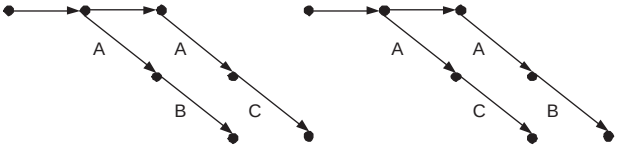

Figure 7: Problematic examples for defining a bisimulation-like formulation of the change semantics

3.3 Semantics with invisible silent steps

We also explore three types of semantics where τ steps cannot be observed: *eager semantics*, *far-sighted semantics*, and *near-sighted semantics*. For each of these three semantical definitions, the observable traces do not show τ steps.

3.3.1 Eager semantics

In the eager semantics the system gives priority to τ steps, i.e., as long as τ steps are possible no offer is made to the environment. For example, T_7, T_9, T_{10}, T_{12} have two possible traces: $\langle\{A, B\}, A, a, \emptyset\rangle$ and $\langle\{A, B\}, B, b, \emptyset\rangle$. T_8 also has two possible traces: $\langle\{A\}, A, a, \emptyset\rangle$ and $\langle\{B\}, B, b, \emptyset\rangle$ and T_{11} only has one possible trace: $\langle\{B\}, B, b, \emptyset\rangle$. Note that in T_{11} a is never offered. One of the problems of the eager semantics is *divergence*, i.e., for some AWs it is possible to have an infinite path of τ steps in the tree. This may lead to the situation that the workflow does not make an offer, not even the empty one.

We now proceed with the formal definition of these semantics.

Definition 3.7 (Eager Workflow Trace). The *set of eager workflow traces* of a CW (V, E, r) is defined as a set of lists of the form $(2^{\mathcal{A}} \cdot \mathcal{A} \cdot \mathcal{I})^* \cdot 2^{\mathcal{A}}$ such that the list $\langle W_1, a_1, i_1, \ldots, W_k, a_k, i_k, W_{k+1}\rangle$ is in this set iff there is a path $n_1 \Rightarrow n_1' \stackrel{(a_1,i_1)}{\rightarrow} n_2 \Rightarrow n_2' \stackrel{(a_2,i_2)}{\rightarrow} n_3 \Rightarrow \ldots n_k' \stackrel{(a_k,i_k)}{\rightarrow} n_{k+1} \Rightarrow n_{k+1}'$ such that $r = n_1$ and for each $1 \leq j \leq k+1$ it holds that $W_j = W(n_j')$ and there is no edge with label $(\tau, \bot)$ that leaves from n_j'.

Based upon these traces we then define the equivalence relation $\equiv_{es}$ as in Definition 2.6 but replacing the original workflow traces with eager workflow

[2] Note that $T_7 \equiv_{nes} T_{12}$, i.e., the trace sets of T_7 and T_{12} coincide because the initial state of T_{12} is invisible.

traces. Under these semantics it holds for example that $T_7 \equiv_{es} T_9$.

3.3.2 Far-sighted semantics

In the far-sighted semantics the system looks at all states that can be reached through zero or more τ steps and offers all activity identifiers of edges that leave from such states. For example, T_8 has two possible traces: $\langle\{A, B\}, A, a, \emptyset\rangle$ and $\langle\{A, B\}, B, b, \emptyset\rangle$.

Formally we can define this as follows. Given a CW (V, E, r) we let $W^*(n)$ denote the *far-sighted work-set* of n, i.e., $W^*(n)$ is the set of all activity identifers a such that there is a path $n \Rightarrow n' \stackrel{(a,i)}{\rightarrow} n''$ in E and $a \neq \perp$.

Definition 3.8 (Far-sighted Workflow Trace). The *set of far-sighted workflow traces* of a CW (V, E, r) is defined as a set of lists of the form $(2^{\mathcal{A}} \cdot \mathcal{A} \cdot \mathcal{I})^* \cdot 2^{\mathcal{A}}$ such that $\langle W_1, a_1, i_1, \ldots, W_k, a_k, i_k, W_{k+1}\rangle$ is in this set iff there is a path $n_1 \Rightarrow n'_1 \stackrel{(a_1,i_1)}{\rightarrow} n_2 \Rightarrow n'_2 \stackrel{(a_2,i_2)}{\rightarrow} n_3 \Rightarrow \ldots n'_k \stackrel{(a_k,i_k)}{\rightarrow} n_{k+1}$ such that $r = n_1$ and for each $1 \leq j \leq k+1$ it holds that $W_j = W^*(n_j)$ and there is no edge with label $(\tau, \perp)$ that arrives in n_j.

As before we then define the equivalence relation $\equiv_{fss}$ as in Definition 2.6 but replacing the original workflow traces with far-sighted workflow traces. Note that the trace sets of all six CWs shown in Figure 6 are identical under the far-sighted semantics.

It is not hard to see that we obtain the same semantics if we omit from the full-semantics all the work-sets except the final work-set, i.e., at the level of CWs the far-sighted semantics is equivalent with the trace semantics in which the τ steps are ignored. Note that this does not mean that the "moment of choice" is no longer captured since such semantics at the level of CWs would still lead to a semantics at the level of AWs that is a generalization of bisimulation.

3.3.3 Near-sighted semantics

In the near-sighted semantics the system can make a choice between making an offer or taking a τ step. If the current state has only τ steps then the system picks one of the τ steps. If there are also visible steps available then the system either makes an offer that corresponds to the edges (labeled with visible steps) leaving from this state and waits for the environment to respond, or picks one of the τ steps. For example, T_9 has three possible traces: $\langle\{A\}, A, a, \emptyset\rangle$ (i.e., the system only offers a), $\langle\{A, B\}, A, a, \emptyset\rangle$, and $\langle\{A, B\}, B, b, \emptyset\rangle$. T_{11} also has three possible traces: $\langle\{A, B\}, A, a, \emptyset\rangle$, $\langle\{A, B\}, B, b, \emptyset\rangle$, and $\langle\{B\}, B, b, \emptyset\rangle$ (i.e., the system only offers b).

Formally we can define these semantics as follows.

Definition 3.9 (Near-sighted Workflow Trace). The *set of near-sighted workflow traces* of a CW (V, E, r) is a set of lists of the form $(2^{\mathcal{A}} \cdot \mathcal{A} \cdot \mathcal{I})^* \cdot 2^{\mathcal{A}}$ such that $\langle W_1, a_1, i_1, \ldots, W_k, a_k, i_k, W_{k+1}\rangle$ is in this set iff there is a path $n_1 \Rightarrow n'_1 \stackrel{(a_1,i_1)}{\rightarrow} n_2 \Rightarrow n'_2 \stackrel{(a_2,i_2)}{\rightarrow} n_3 \Rightarrow \ldots n'_k \stackrel{(a_k,i_k)}{\rightarrow} n_{k+1} \Rightarrow n'_{k+1}$ such that $r = n_1$ and for each $1 \leq j \leq k+1$ it holds that $W_j = W(n'_j)$ and $(a_j, i_j) \neq (\tau, \perp)$.

We then define the equivalence relation $\equiv_{nss}$ as in Definition 2.6 but replacing the original workflow traces with near-sighted workflow traces.

3.4 Summary and related work

Table 1 summarizes our findings. It lists the equivalence classes induced by each of the equivalence notions previously discussed on the set of examples of Figure 6. The purpose of the table is to illustrate that the six proposed equivalence notions are different from each other and different from the two equivalence notions introduced in Section 3.1 (weak and branching bisimilarity). It should be noted that the table suggests that the eager semantics coincides with weak and branching bisimilarity. However, this is not generally true. For instance, consider the AW T'_9 obtained by changing the first "A, a" into a "C, c" in T_9. It can be seen that $T_9 \equiv_{es} T'_9$ but $T_9 \not\equiv_{bbs} T'_9$. Indeed, in the eager semantics the τ step would always be taken before an offer is made, and thus, it is irrelevant whether the alternative transition is labeled A, a or C, c. Also, weak and branching bissimilarity coincide on the six working examples, but it is well known that these notions differ (van Glabbeek & Weijland 1996). Hence, all the equivalence notions listed in Table 1 are distinct.

In the literature, many equivalence notions have been defined (van Glabbeek 1993, van Glabbeek & Weijland 1996). Some of them are similar to our notions of equivalence, e.g., the readiness semantics defined by Olderog and Hoare (Olderog & Hoare 1986) is close to the eager semantics. In (Ingólfsdóttir 1997), a bisimulation-like characterization of readiness semantics for non-divergent processes is provided, under the assumption that taking a τ step never causes a task to be removed from the offered work-set. Also, parallels can be drawn between the treatment of τ steps in the full semantics, and the notion of *autonomous actions* discussed in (Voorhoeve & Basten 1996). In this reference, autonomous actions are defined as steps of a process that may be observed but not controlled by the environment, which captures the intuition that in the full semantics τ steps are internal actions that can be observed by the environment whether they change the work-set or not.

4 Conclusion

Which of the eight equivalence notions presented in Table 1 is most suitable for workflow systems? The two classical ones (trace semantics and branching bisimilarity) are less suitable because they do not consider explicit offers. Of the three equivalence notions where τ steps may be visible (full, change, and non-empty semantics), the change semantics seems the most realistic one since it captures the intuition that the work-set is all that the environment can perceive in-between two inputs. Of the three equivalence notions where τ steps are invisible (eager, far-sighted, and near-sighted semantics), the near-sighted semantics seems to be the most realistic one. Indeed, the eager semantics may create "dead branches" in the transition tree while the far-sighted semantics lets the choice of the τ steps be influenced by the environment, contradicting the assumption that τ steps are entirely controlled by the system.

This paper provides a starting point for defining equivalence relations for workflows. We do not provide a definitive answer to the question raised above. Instead, we have provided a number of formal notions (e.g., AWs, CWs) and raised key issues that need to be considered when choosing an equivalence notion. In addition, we have formalized a notion of equivalence for workflows in the absence of silent steps.

This work could be further pursued in both theoretical and practical directions. From a theoretical perspective, interesting issues include studying the

weak bisimilarity	$\equiv_{wbs}$	$\{T_7, T_9, T_{10}, T_{12}\}, \{T_8\}, \{T_{11}\}$
branching bisimilarity	$\equiv_{bbs}$	$\{T_7, T_9, T_{10}, T_{12}\}, \{T_8\}, \{T_{11}\}$
full semantics	$\equiv_{fs}$	$\{T_7\}, \{T_8\}, \{T_9\}, \{T_{10}\}, \{T_{11}\}, \{T_{12}\}$
change semantics	$\equiv_{cs}$	$\{T_7, T_{10}\}, \{T_8\}, \{T_9\}, \{T_{11}\}, \{T_{12}\}$
non-empty semantics	$\equiv_{nes}$	$\{T_7, T_{12}\}, \{T_8\}, \{T_9\}, \{T_{10}\}, \{T_{11}\}$
eager semantics	$\equiv_{es}$	$\{T_7, T_9, T_{10}, T_{12}\}, \{T_8\}, \{T_{11}\}$
far-sighted semantics	$\equiv_{fss}$	$\{T_7, T_8, T_9, T_{10}, T_{11}, T_{12}\}$
near-sighted semantics	$\equiv_{nss}$	$\{T_7, T_{10}, T_{12}\}, \{T_8\}, \{T_9\}, \{T_{11}\}$

Table 1: Equivalences for the AWs corresponding to the CWs of Figure 6 under various notions of equivalence

properties of the notions of equivalence introduced and in particular, characterizing classes of workflows for which these notions of equivalence (in particular those involving invisible silent steps) correspond with well-understood notions of equivalence such as weak and branching bisimulation. From a practical perspective, the notions of equivalence presented could be applied to proving the correctness of mappings between workflow modeling languages (or proving the impossibility of defining full mappings between certain languages). This is relevant in light of the emergence of proposed standards for business process execution that support silent steps, most notably BPEL (Andrews et al. 2003).

Acknowledgements. This work is partially funded by an ARC Discovery Grant "Expressiveness Comparison and Interchange Facilitation between Business Process Execution Languages".

References

van der Aalst, W. (1998), 'The Application of Petri Nets to Workflow Management', *The Journal of Circuits, Systems and Computers* **8**(1), 21–66.

van der Aalst, W. & Basten, T. (2002), 'Inheritance of Workflows: An Approach to Tackling Problems Related to Change', *Theoretical Computer Science* **270**(1-2), 125–203.

van der Aalst, W. & ter Hofstede, A. (2003), 'YAWL: Yet Another Workflow Language', Accepted for publication in *Information Systems*, and also available as QUT Technical report, FIT-TR-2003-04, Queensland University of Technology, Brisbane.

van der Aalst, W., ter Hofstede, A., Kiepuszewski, B. & Barros, A. (2003), 'Workflow Patterns', *Distributed and Parallel Databases* **14**(1), 5–51.

Andrews, T., Curbera, P., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I. & Weerawarana, S. (2003), 'Business process execution language for web services version 1.1'. Accessed 5 May 2004 from `http://www-106.ibm.com/developerworks/webservices/library/ws-bpel`.

Baeten, J. & Weijland, W. (1990), *Process Algebra*, Vol. 18 of *Cambridge tracts in theoretical computer science*, Cambridge University Press, Cambridge.

Basten, T. (1996), 'Branching Bisimilarity is an Equivalence indeed!', *Information Processing Letters* **58**(3), 141–147.

Engelfriet, J. (1985), 'Determinacy → (observation equivalence = trace equivalence)', *Theoretical Computer Science* **36**, 21–25.

van Glabbeek, R. (1993), The Linear Time - Branching Time Spectrum II: The Semantics of Sequential Systems with Silent Moves, *in* E. Best, ed., 'Proceedings of CONCUR 1993', Vol. 715 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 66–81.

van Glabbeek, R. & Weijland, W. (1996), 'Branching Time and Abstraction in Bisimulation Semantics', *Journal of the ACM* **43**(3), 555–600.

Hollingsworth, D. (1995), Workflow Management Coalition – The Workflow Reference Model, Document number WFMC-TC-1003, Workflow Management Coalition, UK.

Ingólfsdóttir, A. (1997), Weak semantics based on lighted button pressing experiments, *in* 'Proceedings of the 10th International Workshop on Computer Science Logic (CSL), Utrecht, The Netherlands, September 1996', Springer Verlag, pp. 226–243.

Jablonski, S. & Bussler, C. (1996), *Workflow Management: Modeling Concepts, Architecture, and Implementation*, International Thomson Computer Press, London, UK.

Janssens, G., Verelst, J. & Weyn, B. (1998), Reuse-oriented workflow modelling with Petri nets, *in* W. M. P. van der Aalst, ed., 'Proceedings of the Workshop on Workflow Management at the 19th International Conference on Application and Theory of Petri Nets', Lisboa, pp. 40–59.

Kiepuszewski, B., ter Hofstede, A. & van der Aalst, W. (2003), 'Fundamentals of Control Flow in Workflows', *Acta Informatica* **39**(3), 143–209.

Milner, R. (1980), *A Calculus of Communicating Systems*, Vol. 92 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin.

Olderog, E. & Hoare, C. (1986), 'Specification-Oriented Semantics for Communicating Processes', *Acta Informatica* **23**(1), 9–66.

Verelst, J. (2004), A framework for classifying variability in conceptual models, Technical Report RPS-2004-019, University of Antwerp, Dept. of Management Information Systems.

Voorhoeve, M. & Basten, T. (1996), Process algebra with autonomous actions, Technical report 96/01, Eindhoven University of Technology, Department of Mathematics and Computing Science, Eindhoven, the Netherlands.

Wombacher, A. & Mahleko, B. (2002), Finding trading partners to establish ad-hoc business processes, *in* 'On the Move to Meaningful Internet Systems, 2002 - Proceedings of the DOA/CoopIS/ODBASE Confederated International Conferences, Irvine CA, USA', Springer Verlag, pp. 339–355.