Towards approximate equivalences of workflow processes

Yongzhi Cao[1,2] **Guoqing Chen**[3]

[1]Institute of Software, School of Electronics Engineering and Computer Science
Peking University, Beijing 100871, China
[2]Key Laboratory of High Confidence Software Technologies (Peking University)
Ministry of Education, China. E-mail: caoyz@pku.edu.cn
[3]School of Economics and Management, Tsinghua University, Beijing 100084
China. E-mail: chengq@sem.tsinghua.edu.cn

Abstract

Well-known notions of equivalence such as trace equivalence and bisimulation have been used to compare workflow processes; they describe the equivalence of processes whose activities are identical in each step. Possessing exactly the same activity in each step is quite rigorous, and sometimes we may meet processes which fail to fit this condition but which have quite similar behavior. To characterize this kind of looser equivalences, we introduce an approximate version of trace equivalence and bisimulation in this paper.

Keywords: Workflow process, trace equivalence, bisimulation, approximate equivalence

1. Introduction

Workflows (also known as workflow processes or business processes) are activities involving the coordinated execution of single activities (task or worktask) performed by different processing entities. In the last two decades, workflow system ability of providing an automated support to the management of business processes has increasingly become well recognized as a competitive factor for a company [1]. With rapid increases in application domains that use workflow management systems, a large number of workflow modeling languages such as Petri nets, process algebra, BPEL, and YAWL have been developed. Even in a given modeling language, it is often possible to define multiple models of the same workflow. Given the co-existence of different modeling languages and different models of a workflow, there is a need for some formal techniques that can be used to compare these process models.

In the literature, some well-known notions of equivalence such as trace equivalence [2], strong and weak bisimulations [3, 4], and branching bisimulation [3, 5] have been used to compare workflow processes. These equivalences aim to answer when two workflows are the same with respect to some criterion; for example, two workflows may be identical under trace equivalence but are different when considering stronger notions of equivalence such as bisimulation. It is worth noting that most equivalence notions provide a binary answer (i.e., two workflows are equivalent or not). It has been argued by Alves de Medeiros et al. [6]

that these exact equivalences seem too rigid since in real-life applications, one needs to differentiate between slightly different models and completely different models. As a result, they used the degree of similarity, a number between 0 and 1 to quantify the differences of two workflows. Notably, rather than directly comparing two workflows, the processes are compared with respect to some typical behaviors.

In concurrency theory, different process models that cannot be related by behavioral equivalences are typically compared through approximation of some equivalence, which usually gives rise to a quantitative notion of similarity (see, for example, [7, 8, 9, 10, 11]). In this paper, we pursue the approximate equivalence approach by introducing the notions of λ-approximate trace equivalence and λ-approximate bisimulation. Roughly speaking, the exact trace equivalence and bisimulation express the equivalence of workflow processes whose activities are identical in each step. The approximate versions try to compare two workflow processes which are more or less similar in the sense that whenever a process makes an activity the other one can make an activity different from but very similar to the activity the first process made. The smaller the value of λ, the higher the degree to which the two workflow processes are equivalent.

2. Workflows

In this section, we introduce the notions of abstract workflow and concrete workflow, which are a slight extension of the ones in [3]. Like [3], we focus on the workflows abstracted at the level of control flow. Recall that in [3], the behavior of a workflow is described by a *transition tree*, which is a connected, rooted, edge-labeled, and directed graph without cycles. Since the transition tree having no cycles is unequal to dealing with iterative or arbitrarily long behavior, it is not capable of representing the behavior of replication operator in [13]. In light of this, we would like to describe the behavior of a workflow with a rooted labeled transition system, i.e., a rooted, connected, edge-labeled, and directed graph. Formally, we extend the corresponding definitions in [3] as follows.

Definition 1. An *abstract workflow* (or AW for short) is defined as a quadruple $(S, \mathcal{A}, \longrightarrow, s_0)$, where

(1) S is a (possibly infinite) set of states.
(2) $\mathcal{A}$ is a (possibly infinite) set of activity identifiers.
(3) $\longrightarrow \subseteq S \times \mathcal{A} \times S$ is a transition relation.
(4) s_0, a member of S, is the initial state.

The activity identifiers in $\mathcal{A}$ such as "Check credit", "Receive goods", and "Get patient information", provide the activities offered to the environment which may consist of users and applications that interact with the workflow system. As usual, we use the more intuitive notation $s \xrightarrow{a} s'$ instead of $(s, a, s') \in \longrightarrow$. Clearly, Definition 1 is independent of the language used to describe workflows.

As explained in [3], an abstract workflow abstracts from the data that is supplied by the external environment. Therefore, it is necessary to introduce the notion of concrete workflow, which takes this information into account by labeling the edges with a pair of activity identifier and externally supplied information.

Definition 2. A *concrete workflow* (or CW for short) is defined as a five-tuple $(S, \mathcal{A}, \mathcal{I}, \longrightarrow, s_0)$, where

(1) S is a (possibly infinite) set of states.
(2) $\mathcal{A}$ is a (possibly infinite) set of activity identifiers.
(3) $\mathcal{I}$ is a (possibly infinite) set of input data.
(4) $\longrightarrow \subseteq S \times (\mathcal{A} \times \mathcal{I}) \times S$ is a transition relation.
(5) s_0, a member of S, is the initial state.

The input data in $\mathcal{I}$ 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 value such as for the activity "Check credit" or a complex data structure such as for "Get patient information". As before, we use the more intuitive notation $s \xrightarrow{(a,i)} s'$ instead of $(s, (a, i), s') \in \longrightarrow$.

3. Exact equivalences

In this section, we recall the notions of exact trace equivalence and bisimulation. A reasonable first attempt at defining a behavioral equivalence for workflow processes might be to relate two system descriptions exactly when the rooted labeled transition systems for them have exactly the same traces (see, for example, [2]). Before formalizing these notions we first review some concepts from the theory of finite strings. If $\mathcal{A}$ is a set, then $\mathcal{A}^*$ consists of the set of (possibly empty) finite strings of elements of $\mathcal{A}$. We use ϵ to represent the empty string. Let us define traces (cf. [3]) and trace equivalence as follows.

Definition 3. Let $W = (S, \mathcal{A}, \longrightarrow, s_0)$ be an AW and $s = a_1 a_2 \cdots a_n \in \mathcal{A}^*$ be a string of activity identifiers. We say that σ is a *trace* of the AW if there are states $s_1, s_2, \ldots, s_n$ such that $s_{i-1} \xrightarrow{a_i} s_i$ for each $1 \leq i \leq n$. Denote by $\mathcal{T}(W)$ the set of all traces of W. Two AWs W_1 and W_2 are said to be *trace equivalent,* denoted $W_1 \approx W_2$, if $\mathcal{T}(W_1) = \mathcal{T}(W_2)$.

Unfortunately, as is often argued in concurrency theory, trace equivalence suffers from severe deficiencies for workflow processes due to nondeterminism [3]. The trouble with trace equivalence and non-determinism is that even though two AWs have the same traces, they may go through inequivalent states in performing them. (This situation cannot occur in deterministic systems.) In particular, trace equivalent systems may have different deadlocking behavior. This observation suggests that an appropriate equivalence for workflow processes ought to have a recursive flavor: execution sequences for equivalent systems ought to "pass through" equivalent states. This intuition underlies the definition of bisimulation (cf. [3]).

Definition 4. Let $W_i = (S_i, \mathcal{A}, \longrightarrow, s_{i0})$ be an AW, where $i = 1, 2$. A relation $R \subseteq S_1 \times S_2$ is called a *bisimulation* if for any $(s_1, s_2) \in R$ and $a \in \mathcal{A}$,

(1) $s_1 \xrightarrow{a} s_1'$ implies $s_2 \xrightarrow{a} s_2'$ for some s_2' with $(s_1', s_2') \in R$;
(2) $s_2 \xrightarrow{a} s_2'$ implies $s_1 \xrightarrow{a} s_1'$ for some s_1' with $(s_1', s_2') \in R$.

Bisimilarity, denoted $\sim$, is the union of all bisimulations. Two states are called *bisimilar* if they are related by a bisimulation. Two AWs W_1 and W_2 are said to be *bisimilar,* denoted $W_1 \sim W_2$, if their initial states are bisimilar.

Intuitively, if two workflows are bisimilar, then it is possible for each

to simulate, or "track", the other's behavior. More specifically, for a relation to be a bisimulation, related states must be able to "match" transitions of each other by moving to related states. Bisimulation has some pleasing properties. For example, two bisimilar workflows must have the same "deadlock potential"; $\sim$ implies $\approx$ and coincides with it if the rooted labeled transition systems for them are deterministic in the sense that every state has at most one outgoing transition per activity.

Remark 1. Definitions 3 and 4 for AWs can be directly applied to CWs by replacing label a with (a, i). We do not state them there due to the lack of space. Moreover, the results established in Sections 4 and 5 are also directly applicable to CWs, so we omit the corresponding results for CWs.

It should be pointed out that the exact trace equivalence and bisimulation are too rigid in the sense that they differentiate similar activities strictly. For instance, consider two CWs W_1 and W_2 that are nearly the same except for one activity: the former has the activity (a, i), while the latter has the activity (a, i'). However similar their input data i and i', it always holds by definition that $W_1 \not\approx W_2$ and $W_1 \not\sim W_2$.

4. Approximate trace equivalence

Taking similarity of elements into account and using metric to describe the similarity are widely recognized in some fields of Computer Science such as metric semantics, process calculus, and pattern recognition (see, for example, [12] and the bibliographies therein). For our purpose, we would like to follow the approach proposed by the Cao and Ying in [12]. Let us start with some basic notions on metric space.

Definition 5. A *(1-bounded) metric space* is a pair (X, d) consisting of a nonempty set X and a function $d : X \times X \longrightarrow [0, 1]$ which satisfies the following conditions:

(M1) $d(x, y) = 0$ if and only if $x = y$,

(M2) $d(x, y) = d(y, x)$ for all $x, y \in X$, and

(M3) $d(x, z) \leq d(x, y) + d(y, z)$ for all $x, y, z \in X$.

The distance $d(x, y)$ measures the similarity between x and y. The less the distance, the more similar the two elements. It is well-known that for any nonempty set X, we can define a metric d, called *discrete metric*, on X. Hence, every nonempty set can be viewed as a metric space.

Let (X, d) be a metric space, $x_0 \in X$, and $\lambda \in [0, 1]$. The set $B(x_0, \lambda) = \{x \in X : d(x_0, x) \leq \lambda\}$ is called the *λ-ball about x_0*; for a subset A of X, by the *λ-ball about A* we mean that the set $B(A, \lambda) = \cup_{x \in A} B(x, \lambda)$. We extend d to a pair x, A, where $x \in X$ and $A \subseteq X$, by defining

$$d(x, A) = \begin{cases} \inf\limits_{a \in A} d(x, a), & \text{if } A \neq \emptyset \\ 1, & \text{if } A = \emptyset. \end{cases}$$

Further, the *Hausdorff metric* $d_H(A, B)$ for a pair $A, B \subseteq X$ is defined as 0 if $A = B = \emptyset$, and otherwise $\max\{\sup\limits_{a \in A} d(a, B), \sup\limits_{b \in B} d(b, A)\}$. The Hausdorff metric is one of the common ways of measuring resemblance between two sets in a metric space; it satisfies (M2) and (M3) in Definition 5, but it does not satisfy (M1) in general.

Let $W = (S, \mathcal{A}, \longrightarrow, s_0)$ be an AW and d be a metric on $\mathcal{A}$ which makes $\mathcal{A}$

into a metric space. We extend naturally d to d' on $\mathcal{A} \cup \{\epsilon\}$ as follows:

$$d'(a,b) = \begin{cases} d(a,b), & \text{if } a,b \in \mathcal{A} \\ 0, & \text{if } a = b = \epsilon \\ 1, & \text{otherwise.} \end{cases}$$

This makes $\mathcal{A} \cup \{\epsilon\}$ into a metric space.

We now endow $\mathcal{A}^*$ with the *Baire metric* induced by d'. Let $s = s_1 s_2 \cdots s_{l(s)}$ and $t = t_1 t_2 \cdots t_{l(t)}$ be two activity strings in $\mathcal{A}^*$, and $l(s,t) = \max\{l(s), l(t)\}$. If $l(s) \neq l(t)$, say $l(s) < l(t)$, we take $s_i = \epsilon$ for each $i > l(s)$. We then define

$$\tilde{d}(s,t) = \sum_{i=1}^{l(s,t)} \frac{1}{2^i} d'(s_i, t_i).$$

It is easy to verify that $\tilde{d}$ does give rise to a metric on $\mathcal{A}^*$. As mentioned earlier, Hausdorff metric does not give rise to a metric space in general. However, if we consider the powerset $\mathcal{P}(\mathcal{A}^*)$ of $\mathcal{A}^*$ with the Hausdorff metric induced by $\tilde{d}$, it follows from Proposition 1 in [12] that $(\mathcal{P}(\mathcal{A}^*), \tilde{d}_H)$ is a metric space. Since $\mathcal{T}(W) \in \mathcal{P}(\mathcal{A}^*)$ for any workflow with the activity identifier $\mathcal{A}$, we may use $\tilde{d}_H$ to measure the similarity of their trace sets. We can now introduce the key notion.

Definition 6. Let $W_i = (S_i, \mathcal{A}, \longrightarrow, s_{i0})$, $i = 1, 2$, be an AW and d a metric on $\mathcal{A}$ which induces the Hausdorff metric $\tilde{d}_H$ on $\mathcal{P}(\mathcal{A}^*)$. Given $\lambda \in [0,1]$, we say that W_1 and W_2 are *λ-approximate trace equivalent,* denoted $W_1 \approx_\lambda W_2$, if $\tilde{d}_H(\mathcal{T}(W_1), \mathcal{T}(W_2)) \leq \lambda$.

Clearly, the less the value λ, the more similar the trace sets of two workflows.

Proposition 1. *Let $W_i = (S_i, \mathcal{A}, \longrightarrow, s_{i0})$, $i = 1, 2, 3$, be an AW and d a metric on $\mathcal{A}$ which induces the Hausdorff metric $\tilde{d}_H$ on $\mathcal{P}(\mathcal{A}^*)$. Then we have the following.*

(1) $W_1 \approx_0 W_2$ if and only if $W_1 \approx W_2$.
(2) $W_1 \approx_\lambda W_2$ if and only if $W_2 \approx_\lambda W_1$.
(3) If $W_1 \approx_\lambda W_2$, then $W_1 \approx_{\lambda'} W_2$ for any $\lambda' \geq \lambda$.
(4) If $W_1 \approx_\lambda W_2$ and $W_2 \approx_{\lambda'} W_3$, then $W_1 \approx_{\lambda+\lambda'} W_3$.

Proof. It is easy by Definition 6 and the properties of metric space. □

The assertion (1) in Proposition 1 shows us that λ-approximate trace equivalence is really a generalization of the exact trace equivalence.

5. Approximate bisimulation

In this section, following [10, 11] we introduce an approximate version of bisimulation for workflow processes. Like the above approximate trace equivalence, we assume that there is a metric d on the activity identifier set $\mathcal{A}$ of an AW.

Definition 7. Let $W_i = (S_i, \mathcal{A}, \longrightarrow, s_{i0})$ be an AW, where $i = 1, 2$, and let d be a metric on $\mathcal{A}$. Given $\lambda \in [0,1]$, a relation $R \subseteq S_1 \times S_2$ is called a *λ-approximate bisimulation* if for any $(s_1, s_2) \in R$ and $a \in \mathcal{A}$,

(1) $s_1 \xrightarrow{a} s_1'$ implies $s_2 \xrightarrow{b} s_2'$ for some s_2' such that $(s_1', s_2') \in R$ and $d(a,b) \leq \lambda$;
(2) $s_2 \xrightarrow{a} s_2'$ implies $s_1 \xrightarrow{b} s_1'$ for some s_1' such that $(s_1', s_2') \in R$ and $d(a,b) \leq \lambda$.

Two states are called *λ-approximately bisimilar* if they are related by a λ-approximate bisimulation. Two AWs W_1 and W_2 are said to be *λ-approximately bisimilar,* denoted $W_1 \sim_\lambda W_2$, if their initial states s_{10} and s_{20} are λ-approximately bisimilar.

Intuitively, if two workflows are λ-approximately bisimilar, then it is possible for each to approximately simulate, or "track", the other's behavior. It should be pointed out that the smaller the value of λ, the higher the degree to which R is a bisimulation.

The following fact indicates that the class of λ-approximate bisimulations is closed under arbitrary unions. In this way, we can get the largest λ-approximate bisimulation.

Proposition 2. *If R_i is a λ_i-approximate bisimulation for any $i \in I$, then $\bigcup_{i\in I} R_i$ is a $\sup\{\lambda_i : i \in I\}$-approximate bisimulation. In particular, if every R_i is a λ-approximate bisimulation, then so is $\bigcup_{i\in I} R_i$.*

Proof. It is straightforward from Definition 7. □

Similar to Proposition 1, λ-approximate bisimulations have the following properties.

Proposition 3. *Let $W_i = (S_i, \mathcal{A}, \longrightarrow, s_{i0})$, $i = 1, 2, 3$, be an AW and d be a metric on $\mathcal{A}$. Then we have the following.*

(1) $W_1 \sim_0 W_2$ if and only if $W_1 \sim W_2$.
(2) $W_1 \sim_\lambda W_2$ if and only if $W_2 \sim_\lambda W_1$.
(3) If $W_1 \sim_\lambda W_2$, then $W_1 \sim_{\lambda'} W_2$ for any $\lambda' \geq \lambda$.
(4) If $W_1 \sim_\lambda W_2$ and $W_2 \sim_{\lambda'} W_3$, then $W_1 \sim_{\lambda+\lambda'} W_3$.

Proof. It is also easy by Definition 7 and the properties of metric space. □

The assertion (1) in Proposition 3 shows that λ-approximate bisimulation has the standard bisimulation as a specific case. Not only that, the notion of instance relation proposed in [3] can also be considered as a λ-approximate bisimulation. To see this, let us recall the notion of instance relation from [3].

Definition 8. An *instance relation* between an AW $(S_1, \mathcal{A}, \longrightarrow, s_{10})$ and a CW $(S_2, \mathcal{A} \times \mathcal{I}, \longrightarrow, s_{20})$ is a relation $H \subseteq S_1 \times S_2$ such that $(s_{10}, s_{20}) \in H$ and for any $(s_1, s_2) \in H$,

(1) if $s_1 \xrightarrow{a} s_1'$ then $s_2 \xrightarrow{(a,i)} s_2'$ for some s_2' with $(s_1', s_2') \in H$;
(2) if $s_2 \xrightarrow{(a,i)} s_2'$ then $s_1 \xrightarrow{a} s_1'$ for some s_1' with $(s_1', s_2') \in H$.

In order to view an instance relation as a λ-approximate bisimulation, let us specify a metric d on $\mathcal{A} \cup (\mathcal{A} \times \mathcal{I})$ as follows. Given $\lambda_0 \in (0, 1)$, for any $(x, y) \in \mathcal{A} \cup (\mathcal{A} \times \mathcal{I})$, define

$$d_0(x,y) = \begin{cases} 0, & \text{if } x = y \\ \lambda_0, & \text{if } x = a \text{ and } y = (a, i), \text{ and vice versa} \\ \lambda_0, & \text{if } x = (a, i),\ y = (a, j),\ i \neq j \\ 1, & \text{otherwise.} \end{cases}$$

There is no difficulty to verify that d_0 is a metric on $\mathcal{A} \cup (\mathcal{A} \times \mathcal{I})$. As a result, we get immediately the following observation.

Proposition 4. *Let H be an instance relation between an AW $(S_1, \mathcal{A}, \longrightarrow, s_{10})$ and a CW $(S_2, \mathcal{A} \times \mathcal{I}, \longrightarrow, s_{20})$. Then H is a λ_0-approximate bisimulation for any $\lambda_0 \in (0, 1)$.*

Acknowledgment

This work was supported by the National Natural Science Foundation of China under Grants 70890080, 60873061, and 60973004 and by the National Basic Research Program of China (973 Program) under Grants 2007CB311003, 2009CB320701, and 2010CB328103.

References

[1] M. Dumas, W. van der Aalst and A. ter Hofstede, *Process-aware information systems: bridging people and software through process technology*, Wiley-Blackwell, 2005.

[2] A. Wombacher and B. Mahleko, Finding trading partners to establish ad-hoc business processes. In *On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE*, Lect. Notes Comput. Sci. 2519, pages 339–355, Springer- Verlag, 2010.

[3] J. Hidders, M. Dumas, W. van der Aalst, A. ter Hofstede and J. Verelst. When are two workflows the same? In *Proc. 2005 Austr. Symp. Theory Comput.- Volume 41*, pages 3–11. Australian Computer Society, Inc., 2005.

[4] B. Kiepuszewski, A. ter Hofstede and W. van der Aalst, Fundamentals of control flow in workflows, *Acta Inform.*, 39(3):143–209, Springer- Verlag, 2003.

[5] W. van der Aalst and T. Basten, Inheritance of workflows: an approach to tackling problems related to change, *Theor. Comput. Sci.*, 270(1-2):125–203, Elsevier, 2002.

[6] A. Alves de Medeiros, W. van der Aalst and A. Weijters, Quantifying process equivalence based on observed behavior, *Data Knowl. Engin.*, 64(1):55–74, Elsevier, 2008.

[7] A. Giacalone, C. Jou and S. Smolka, Algebraic reasoning for probabilistic concurrent systems. In *Proc. Work. Conf. Progr. Conc. Meth., vol. 2 of IFIP*, pages 453–459, Springer, 1990.

[8] A. Girard and G. J. Pappas, Approximation metrics for discrete and continuous systems, *IEEE Trans. Autom. Contr.*, 52(5):782–798, IEEE, 2007.

[9] F. van Breugel and J. Worrell, A behavioural pseudometric for probabilistic transition systems, *Theor. Comput. Sci.*, 331(1):115–142, Elsevier, 2005.

[10] M. Ying, Bisimulation indexes and their applications, *Theor. Comput. Sci.*, 275(1-2):1–68, Elsevier, 2002.

[11] M. Ying and M. Wirsing, Approximate bisimilarity. In Teodor Rus, editor, *Proc. 8th Intern. Conf. Algebraic Methodology and Software Technology*, AMAST 2000, Lect. Notes Comput. Sci. 1816, pages 309–322, Springer, 2000.

[12] Y. Cao and M. Ying, Similarity-based supervisory control of discrete-event systems, *IEEE Trans. Autom. Contr.*, 51(2):325–330, IEEE, 2006.

[13] F. Puhlmann and M. Weske, Using the π-calculus for formalizing workflow patterns, In *Business Process Management*, Lect. Notes Comput. Sci. 3649, pages 153–168, Springer, 2005.

[14] E. De Maria, A. Montanari and M. Zantoni, An automaton-based approach to the verification of timed workflow schemas. In *Proc. Thirteenth Intern. Symp. Tempor. Repr. Reasoning*, TIME'06, pages 87–94, 2006.

[15] W. van der Aalst, T. Weijters and L. Maruster, Workflow mining: Discovering process models from event logs, *IEEE Trans. Knowl. Data Engin.*, pages 1128–1142, IEEE, 2004.