mathematics

MDPI

Article

Knowledge Dynamics and Behavioural Equivalences in Multi-Agent Systems

Bogdan Aman [1,2] and Gabriel Ciobanu [2,*]

[1] Institute of Computer Science, Romanian Academy, 700505 Iaşi, Romania; bogdan.aman@iit.academiaromana-is.ro
[2] Faculty of Computer Science, Alexandru Ioan Cuza University, 700506 Iaşi, Romania
* Correspondence: gabriel@info.uaic.ro

Abstract: We define a process calculus to describe multi-agent systems with timeouts for communication and mobility able to handle knowledge. The knowledge of an agent is represented as sets of trees whose nodes carry information; it is used to decide the interactions with other agents. The evolution of the system with exchanges of knowledge between agents is presented by the operational semantics, capturing the concurrent executions by a multiset of actions in a labelled transition system. Several results concerning the relationship between the agents and their knowledge are presented. We introduce and study some specific behavioural equivalences in multi-agent systems, including a knowledge equivalence able to distinguish two systems based on the interaction of the agents with their local knowledge.

Keywords: mobile agents; timeouts; knowledge as set of trees; behavioural equivalences

Citation: Aman, B.; Ciobanu, G. Knowledge Dynamics and Behavioural Equivalences in Multi-Agent Systems. *Mathematics* **2021**, *9*, 2869. https://doi.org/10.3390/math9222869

Academic Editor: Liliya Demidova

Received: 25 September 2021
Accepted: 8 November 2021
Published: 11 November 2021

1. Introduction

Process calculi are used to describe concurrent systems, providing a high-level description of interactions, communications and synchronizations between independent processes or agents. The main features of a process calculus are: (i) interactions between agents/processes are by communication (message-passing), rather than modifying shared variables; (ii) large systems are described in a compositional way by using a small number of primitives and operators; (iii) processes can be manipulated by using equational reasoning and behavioural equivalences. The key primitive distinguishing the process calculi from other models of computation is the parallel composition. The compositionality offered by the parallel composition can help to describe large systems in a modular way, and to better organize their knowledge (for reasoning about them).

In this paper we define an extension of the process calculus TIMO [1] in order to model multi-agent systems and their knowledge. In this framework, the agents can move between locations and exchange information, having explicit timeouts for both migration and communication. Additionally, they have a knowledge of the network used to decide the next interactions with other agents. The knowledge of the agents is inspired by a model of semi-structured data [2] in which it is given by sets of unordered trees containing pairs of labels and values in each node. In our approach, the knowledge is described via sets of trees used to exchange information among agents about migration and communication. Overall, we present a formal way to describe the behaviour of mobile communicating agents and networks of agents in a compositional manner.

A network of mobile agents is a distributed environment composed of locations where several agents act in parallel. Each agent is represented by a process together with its knowledge that is used to decide interactions with other agents. Taking the advantage that there already exists a theory of parallel and concurrent systems, we define a prototyping language for multi-agent systems presented as a process calculus in concurrency theory. Its

semantics is given formally by a labelled transition system; in this way we describe the behaviour of the entire network, and prove some useful properties.

In concurrency, the behavioural equality of two systems is captured by using bisimulations. Bisimulations are important contributions to computer science that appeared as refinements of 'structure-preserving' mappings (morphisms) in mathematics; they can be applied to new fields of study, including multi-agent systems. Bisimilarity is the finest behavioural equivalence; it abstracts from certain details of the systems, focusing only on the studied aspects. The equivalence relations should be compositional such that if two systems are equivalent, then the systems obtained by their compositions with a third system should also be equivalent. This compositional reasoning allows for the development of complex systems in which each component can be replaced by an equivalent one. Furthermore, there exist efficient algorithms for bisimilarity checking and compositionality properties of bisimilarity, algorithms that are usually used to minimize the state-space of systems. These are good reasons why we consider that it is important to define and study some specific behavioural equivalences for multi-agent systems enhanced with a knowledge of the network for deciding the next interactions. To be more realistic, we consider systems of agents with timing constraints on migration and communication. Therefore, a notable advantage of using our framework to model systems of mobile agents is the possibility to naturally express compositionality, mobility, local communication, timeouts, knowledge, and equivalences between systems in a given interval of time (up to a timeout).

The paper is structured as follows: Section 2 presents the syntax and semantics of the new process calculus knowTIMO and provides some results regarding the timing and knowledge aspects of the evolution. In Section 3 we define and study various bisimulations for the multi-agent systems described in knowTIMO. The conclusion, related work and references end the article.

2. The New Process Calculus knowTIMO

In order to model the evolution of multi-agent systems handling knowledge, timed communication and timed migration, we define a process calculus named knowTIMO, where know stands for 'knowledge' and TIMO stands for the family of calculi introduced in [1] and developed in several articles.

In Table 1 we present the syntax of knowTIMO, where:

- $Loc = \{l, l' \ldots\}$ is a set of distributed locations or location variables, $Chan = \{a, b, \ldots\}$ is a set of channels used for communication among agents, $Id = \{id, \ldots\}$ is a set of names used to denote recursive processes, and $\mathcal{N} = \{N, N', \ldots\}$ is a set of networks;
- a unique process definition $id(u_1, \ldots, u_{m_{id}}) \stackrel{def}{=} P_{id}$ is available for all $id \in Id$;
- timeouts of actions are denoted by $t \in \mathbb{N}$; thresholds appearing in tests are denoted by $k \in \mathbb{Z}$; variables are denoted by u; expressions (over values, variables and allowed operations) are denoted by v; fields are denoted by f; path of fields are denoted by p and are used to retrieve/update the value of the fields. Also, if $Q \in Id$ and $Q(u)$ is a process definition, then for $v_1 \neq v_2$ we obtain two different process instances $Q(v_1)$ and $Q(v_2)$.

An agent A is a pair $P \rhd K$, where A behaves as prescribed by P and K is the knowledge used by process P during its execution. An agent $A = \mathsf{go}^t\, l$ then $P \rhd K$ is ready to migrate from its current location to the location l by consuming the action $\mathsf{go}^t\, l$ of agent A. In go^t, the timer t indicates the fact that agent A is unavailable for t units of time at the current location; then, once the timer t expires, $\mathsf{go}^t\, l$ then P executes process P at the new location l. Since l can be a location variable, it may be instantiated after communication between agents. The use of location variables allows agents to adapt their behaviours based on the interactions among agents.

An agent $A = a^{\Delta t}!\langle v \rangle$ then P else $Q \rhd K$ is available for up-to t units of time to communicate on channel a the value v to another agent $A' = a^{\Delta t'}?(u)$ then P' else $Q' \rhd K'$ available for communication at the same location and awaiting for a value on the same communication channel a. In order to simplify the presentation in this paper, we consider a

synchronous calculus; this means that when a communication takes place, the message sent by one process is instantly received by the other process. If the communication happens, then agent A executes process P, while agent A' executes process P' by making use of the received value v. If the timers t and t' of the agents A and A' expire, then they execute processes Q and Q', respectively.

Table 1. Syntax of our Multi-Agent Systems.

Processes	P, Q	::=	$\mathsf{go}^t\ l$ then P	(move)
		∣	$a^{\Delta t}!\langle v\rangle$ then P else Q	(output)
		∣	$a^{\Delta t}?(u)$ then P else Q	(input)
		∣	if *test* then P else Q	(branch)
		∣	0	(termination)
		∣	$id(v)$	(recursion)
		∣	$create(\langle f \mid v; \emptyset\rangle)$ then P	(create)
		∣	$update(p/f, v)$ then P	(update)
Knowledge	K	::=	$\emptyset$	(empty)
		∣	$\langle f \mid \varepsilon; K\rangle$ ∣ $\langle f \mid v; K\rangle$	(tree)
		∣	$K\ K$	(set)
Paths	p, p'	::=	$/f$ ∣ $p\ p'$ ∣ $p[test(p)]$ ∣ $p[test(p/f)]$	
Tests	$test(p)$	::=	$true$ ∣ $\neg test(p)$ ∣ $K(p) > k$ ∣ $K(p) = v$ ∣ ...	
	$test$	::=	$test(p) \wedge test(p')$ ∣ $\neg test$	
Agents	A, B	::=	$P \triangleright K$	
Set of Agents	$\tilde{A}$	::=	$\mathbf{0}$ ∣ $\tilde{A} \mid\mid A$	
Networks	N	::=	$l[[\tilde{A}]]$ ∣ $N \mid N$	

An agent $A =$ if *test* then P else $Q \triangleright K$ uses its knowledge K to check the truth value of the *test*. If the value is *true*, then agent A executes process P, while if the value is *false*, then agent A executes process Q.

The agent $A = create(\langle f \mid v; \emptyset\rangle)$ then $P \triangleright K$ extends its knowledge K by adding the new piece of knowledge $\langle f \mid v; \emptyset\rangle$ in parallel with K, and then executes process P. The agent $A = update(p/f, v)$ then $P \triangleright K$ updates its knowledge K by adding the value v into the field identified by f reached following path p/f, and then executes process P; if the field f does not exist, then the field is created and the value v is assigned to it. The agent $A = 0 \triangleright K$ has no actions to execute, and its evolution terminates.

The knowledge K of an agent A is used either for storing information needed for communication with other agents or for deciding what process to execute. We define the knowledge as sets of trees in which the nodes carrying the information are of two types: $\langle f \mid \varepsilon; K'\rangle$ and $\langle f \mid v; K'\rangle$. Both types of nodes contain a field f and a knowledge K'; they differ only in the value stored in the field f, which can be either the symbol ε indicating the empty value, or a non-empty value v. An agent $A = P \triangleright K$ can use the information stored in its knowledge K to perform tests. For example, a test $K(p/f) > k$ is *true* only if, following a path p in knowledge K, the value stored in the field f is greater than k (otherwise, it is evaluated to *false*); a path is used to select a node in knowledge K. Predicates, always embedded in square brackets and attached to fields in a path, are used to analyze either the value of the current node by using $p[test(p)]$ or the values of the inner nodes by using $p[test(p/f)]$. We say that a knowledge K is included in another knowledge K' (denoted $K \subseteq K'$) if for all paths p appearing in K it holds that $K(p) = K'(p)$.

In Table 1 there exist only one possibility to bind variables; namely, the variable u of the process $a^{\Delta t}?(u)$ then P else Q is bound within process P, while it is not bound within process Q. We denote by $fv(P)$ and $fv(N)$ the sets of free variables appearing in process P and network N, respectively. Moreover, we impose that $fv(P_{id}) \subseteq \{u_1, \ldots, u_{m_{id}}\}$, where $id(u_1, \ldots, u_{m_{id}}) \stackrel{def}{=} P_{id}$. We denote by $\{v/u\}P$ the process P having all the free occurrences of the variable u replaced by value v, possibly after using α-conversion to avoid name clashes in process P.

A network is composed of distributed locations, where $l[[\tilde{A}]]$ denotes a location l containing a set $\tilde{A}$ of agents, while $l[[\mathbf{0}]]$ denotes a location without any agents. Over the set $\mathcal{N}$ of networks we define the structural equivalence $\equiv$ as the smallest congruence satisfying the equalities:

$$l[[\tilde{A} \mid\mid \mathbf{0}]] \equiv l[[\tilde{A}]] \ , \ l[[\tilde{A}]] \mid l[[\tilde{B}]] \equiv l[[\tilde{A} \mid\mid \tilde{B}]] \ ,$$
$$N \equiv N \ , \ N \mid N' \equiv N' \mid N \ , \ (N \mid N') \mid N'' \equiv N \mid (N' \mid N'') \ .$$

The structural congruence $\equiv$ is needed when using the *operational semantics* presented in Tables 2 and 3 for either executing actions or indicating time passing. In Table 2 the relation $N \xrightarrow{\Lambda} N'$ denotes the transformation of a network N into a network N' by executing the actions from the multiset of actions Λ; if the multiset of actions Λ contains only a single action λ, namely $\Lambda = \{\lambda\}$, then we use $N \xrightarrow{\lambda} N'$ instead of $N \xrightarrow{\{\lambda\}} N'$.

The operational semantics of knowTiMo is presented in Table 2.

Table 2. Operational Semantics for our Multi-Agent Systems

(Stop)	$l[[\mathbf{0}]] \not\rightarrow$
(Com)	$l[[a^{\Delta t_1}!\langle v\rangle \text{ then } P_1 \text{ else } Q_1 \rhd K_1 \mid\mid a^{\Delta t_2}?(u) \text{ then } P_2 \text{ else } Q_2 \rhd K_2 \mid\mid \tilde{A}]] \xrightarrow{a!?@l} l[[P_1 \rhd K_1 \mid\mid \{v/u\}P_2 \rhd K_2 \mid\mid \tilde{A}]]$
(Put0)	$l[[a^{\Delta 0}!\langle v\rangle \text{ then } P \text{ else } Q \rhd K \mid\mid \tilde{A}]] \xrightarrow{a!^{\Delta 0}@l} l[[Q \rhd K \mid\mid \tilde{A}]]$
(Get0)	$l[[a^{\Delta 0}?(u) \text{ then } P \text{ else } Q \rhd K \mid\mid \tilde{A}]] \xrightarrow{a?^{\Delta 0}@l} l[[Q \rhd K \mid\mid \tilde{A}]]$
(Move0)	$l[[\text{go}^0\, l' \text{ then } P \rhd K \mid\mid \tilde{A}]] \mid l'[[\tilde{B}]] \xrightarrow{l \triangleright l'} l[[\tilde{A}]] \mid l'[[P \rhd K \mid\mid \tilde{B}]]$
(IfT)	$\dfrac{test@K = true}{l[[\text{if } test \text{ then } P \text{ else } Q \rhd K \mid\mid \tilde{A}]] \xrightarrow{true@l} l[[P \rhd K \mid\mid \tilde{A}]]}$
(IfF)	$\dfrac{test@K = false}{l[[\text{if } test \text{ then } P \text{ else } Q \rhd K \mid\mid \tilde{A}]] \xrightarrow{false@l} l[[Q \rhd K \mid\mid \tilde{A}]]}$
(Create)	$l[[create(\langle f \mid v; \emptyset\rangle) \text{ then } P \rhd K \mid\mid \tilde{A}]] \xrightarrow{create_f@l} l[[P \rhd K\, \langle f \mid v; \emptyset\rangle \mid\mid \tilde{A}]]$
(Update)	$\dfrac{\exists p/f = /f' \ldots /f \qquad K = \langle f' \mid v'; \ldots \langle f \mid v''; K_1\rangle\, K_2\rangle\, K_3 \quad K' = \langle f' \mid v'; \ldots \langle f \mid v; K_1\rangle\, K_2\rangle\, K_3}{l[[update(p/f, v) \text{ then } P \rhd K \mid\mid \tilde{A}]] \xrightarrow{upd_p@l} l[[P \rhd K' \mid\mid \tilde{A}]]}$
(Extend)	$\dfrac{\exists p = /f' \ldots /f'' \qquad \nexists p/f \qquad K = \langle f' \mid v'; \ldots \langle f'' \mid v''; K_1\rangle\, K_2\rangle\, K_3 \quad K' = \langle f' \mid v'; \ldots \langle f'' \mid v''; \langle f \mid v; \emptyset\rangle K_1\rangle\, K_2\rangle\, K_3}{l[[update(p/f, v) \text{ then } P \rhd K \mid\mid \tilde{A}]] \xrightarrow{upd_p@l} l[[P \rhd K' \mid\mid \tilde{A}]]}$
(Call)	$l[[id(v) \rhd K \mid\mid \tilde{A}]] \xrightarrow{call@l} l[[\{v/u\}P_{id} \rhd K \mid\mid \tilde{A}]]$, where $id(u) \stackrel{def}{=} P_{id}$
(Par)	$\dfrac{N_1 \xrightarrow{\Lambda_1} N_1' \quad N_2 \xrightarrow{\Lambda_2} N_2'}{N_1 \mid N_2 \xrightarrow{\Lambda_1 \mid \Lambda_2} N_1' \mid N_2'}$ (Equiv) $\dfrac{N \equiv N' \quad N' \xrightarrow{\Lambda} N'' \quad N'' \equiv N'''}{N \xrightarrow{\Lambda} N'''}$

In rule (Stop), $l[[\mathbf{0}]]$ denotes a network without agents, and thus $\not\rightarrow$ marks the fact that no action is available for execution. Rule (Com) is used if at location l two agents $A_1 = a^{\Delta t_1}!\langle v\rangle$ then P_1 else $Q_1 \rhd K_1$ and $A_2 = a^{\Delta t_2}?(u)$ then P_2 else $Q_2 \rhd K_2$ can communicate successfully over channel a. After communication, both agents remain at the current location l with their knowledge unchanged; agent A_1 executes P_1, while agent A_2 exe-

cutes $\{v/u\}P_2$. The successful communication over channel a at location l is marked by label $a!?@l$.

Rules (PUT0) and (GET0) are used for an agent $A = a^{\Delta 0} * \text{ then } P \text{ else } Q \rhd K$ (where $* \in \{!\langle v\rangle, ?(u)\}$) to remove action a when its timer expires. Afterwards, agent A is ready to execute Q. Knowledge K remains unchanged. Since rule (COM) can be applied even if t_1 and t_2 are zero, it follows that when a timer is 0, only one of the rules (COM), (PUT0) and (GET0) is chosen for application in a nondeterministic manner.

Rule (MOVE0) is used when at location l an agent $A = \mathsf{go}^0\, l' \text{ then } P \rhd K$ migrates to location l' to execute process P. Rules (IFT) and (IFF) are used when an agent $A = \text{if } test \text{ then } P \text{ else } Q \rhd K$ should decide what process to execute (P or Q) based on the Boolean value returned by $test@K$; this value is determined by performing the *test* on the knowledge K of agent A. Notice that in order to perform a *test*, the agent A can only read its knowledge K.

Rule (CREATE) is used when an agent $A = create(\langle f \mid v; \emptyset\rangle) \text{ then } P \rhd K$ extends its knowledge K with $\langle f \mid v; \emptyset\rangle$; afterwards, the agent A executes process P.

Rule (UPDATE) is used when an agent $A = update(p/f, v) \text{ then } P \rhd K$ updates to v the value of $K(p/f)$ of the existing field f, while rule (EXTEND) is used when the agent $A = update(p/f, v) \text{ then } P \rhd K$ expands (at the end of) an existing path p with a field f such that $K(p/f) = v$; afterwards the agent A executes process P.

Rule (CALL) is used when an agent $A = id(v) \rhd K$ is ready to unfold the process $id(v)$ into $\{v/u\}P_{id}$. Rule (PAR) is used to put together the behaviour of smaller subnetworks. while rule (EQUIV) is used to apply the structures congruence over networks.

In Table 3 are presented the rules for describing time passing, while the knowledge of the involved agents remains unchanged. The relation $N \overset{t}{\rightsquigarrow} N'$ indicates the transformation of a network N into a network N' after t units of time.

Table 3. Operational Semantics of knowTIMO : Time Passing.

(DSTOP)	$l[[\mathbf{0}]] \overset{t}{\rightsquigarrow} \mathbf{1}[[\mathbf{0}]]$
(DPUT)	$\dfrac{t \geq t' \geq 0}{l[[a^{\Delta t}!\langle v\rangle \text{ then } P \text{ else } Q \rhd K]] \overset{t'}{\rightsquigarrow} l[[a^{\Delta t - t'}!\langle v\rangle \text{ then } P \text{ else } Q \rhd K]]}$
(DGET)	$\dfrac{t \geq t' \geq 0}{l[[a^{\Delta t}?(u) \text{ then } P \text{ else } Q \rhd K]] \overset{t'}{\rightsquigarrow} l[[a^{\Delta t - t'}?(u) \text{ then } P \text{ else } Q \rhd K]]}$
(DMOVE)	$\dfrac{t \geq t' \geq 0}{l[[\mathsf{go}^t\, l' \text{ then } P \rhd K]] \overset{t'}{\rightsquigarrow} l[[\mathsf{go}^{t-t'}\, l' \text{ then } P \rhd K]]}$
(DPAR)	$\dfrac{N_1 \overset{t}{\rightsquigarrow} N_1' \qquad N_2 \overset{t}{\rightsquigarrow} N_2' \qquad N_1 \mid N_2 \not\rightarrow}{N_1 \mid N_2 \overset{t}{\rightsquigarrow} N_1' \mid N_2'}$
(DEQUIV)	$\dfrac{N \equiv N' \quad N' \overset{t}{\rightsquigarrow} N'' \quad N'' \equiv N'''}{N \overset{t}{\rightsquigarrow} N'''}$

In rule (STOP), $l[[\mathbf{0}]]$ denotes a network without agents; the passing of time does not affect such a network. Rules (DPUT), (DGET) and (DMOVE) are used to decrease the timers of actions, while rules (DPAR) is used to put together the behaviour of composed networks. In rule (DPAR), $N_1 \mid N_2 \not\rightarrow$ denotes a network $N_1 \mid N_2$ that cannot execute any action; this is possible because the use of negative premises in our operational semantics does not lead to inconsistencies.

Given a finite multiset of actions $\Lambda = \{\lambda_1, \ldots, \lambda_k\}$ and a timeout t, a derivation $N \stackrel{\Lambda,t}{\Longrightarrow} N'$ captures a complete computational step of the form:

$$N \stackrel{\lambda_1}{\longrightarrow} N_1 \ldots N_{k-1} \stackrel{\lambda_k}{\longrightarrow} N_k \stackrel{t}{\rightsquigarrow} N'.$$

The fact that a knowTiMo network N is able to perform zero or more actions steps and a time step in order to reach a network N' is denoted by $N \Longrightarrow^* N'$. Notice that the consumed actions and elapsed time are not recorded. By $N \stackrel{\lambda}{\Longrightarrow}^* N'$ we denote the fact that there exist networks N_1 and N_2 such that $N \Longrightarrow^* N_1 \stackrel{\lambda}{\longrightarrow} N_2 \Longrightarrow^* N'$; in this way we emphasize only the consumed action λ out of all consumed actions.

In our setting, at most one time passing rule can be applied for any arbitrary given process. This is the reason why, by inverting a rule, we can describe how the time passes in the subprocesses of a process. This result is useful when reasoning by induction on the structure of processes for which time passes.

Proposition 1. *Assume $N \stackrel{t'}{\rightsquigarrow} N'$. Then exactly one of the following holds:*

- $N = l[[\mathbf{0}]]$ *and* $N' = l[[\mathbf{0}]]$;
- $N = l[[a^{\Delta t}!\langle v\rangle \text{ then } P \text{ else } Q \rhd K]]$ *and* $N' = l[[a^{\Delta t - t'}!\langle v\rangle \text{ then } P \text{ else } Q \rhd K]]$, *where* $t \geq t' \geq 0$;
- $N = l[[a^{\Delta t}?(u) \text{ then } P \text{ else } Q \rhd K]]$ *and* $N' = l[[a^{\Delta t - t'}?(u) \text{ then } P \text{ else } Q \rhd K]]$, *where* $t \geq t' \geq 0$;
- $N = l[[\text{go}^t\, l' \text{ then } P \rhd K]]$ *and* $N' = l[[\text{go}^{t-t'}\, l' \text{ then } P \rhd K]]$, *where* $t \geq t' \geq 0$;
- $N = N_1 \mid N_2$ *such that* $N_1 \mid N_2 \not\rightarrow$, *and there exist* N_1' *and* N_2' *such that* $N' = N_1' \mid N_2'$, $N_1 \stackrel{t'}{\rightsquigarrow} N_1'$ *and* $N_2 \stackrel{t'}{\rightsquigarrow} N_2'$.

Proof. Straightforward, by observing that the time passing rules in Table 3 can be deterministically inverted; namely, each network of Table 1 performing a time step can use at most one rule of Table 3. □

The following theorem claims that time passing does not introduce nondeterminism in the evolution of a network.

Theorem 1. *The next two statements hold for any three networks N, N' and N'':*

1. *if $N \stackrel{0}{\rightsquigarrow} N'$, then $N = N'$;*
2. *if $N \stackrel{t}{\rightsquigarrow} N'$ and $N \stackrel{t}{\rightsquigarrow} N''$, then $N' = N''$.*

Proof. 1. We proceed by induction on the structure of N.

- Case $N = l[[\mathbf{0}]]$. Since $N \stackrel{0}{\rightsquigarrow} N'$, by using Proposition 1, it holds that $N' = l[[\mathbf{0}]]$, meaning that $N = N'$ (as desired).
- Case $N = l[[a^{\Delta t}!\langle v\rangle \text{ then } P \text{ else } Q \rhd K]]$. Since $N \stackrel{0}{\rightsquigarrow} N'$, by using Proposition 1, it holds that $N' = l[[a^{\Delta t - 0}!\langle v\rangle \text{ then } P \text{ else } Q \rhd K]] = l[[a^{\Delta t}!\langle v\rangle \text{ then } P \text{ else } Q \rhd K]]$, meaning that $N = N'$ (as desired).
- Case $N = l[[a^{\Delta t}?(u) \text{ then } P \text{ else } Q \rhd K]]$. Since $N \stackrel{0}{\rightsquigarrow} N'$, by using Proposition 1, it holds that $N' = l[[a^{\Delta t - 0}?(u) \text{ then } P \text{ else } Q \rhd K]] = l[[a^{\Delta t}?(u) \text{ then } P \text{ else } Q \rhd K]]$, meaning that $N = N'$ (as desired).
- Case $N = l[[\text{go}^t\, l' \text{ then } P \rhd K]]$. Since $N \stackrel{0}{\rightsquigarrow} N'$, by using Proposition 1, it holds that $N' = l[[\text{go}^{t-0}\, l' \text{ then } P \rhd K]] = l[[\text{go}^t\, l' \text{ then } P \rhd K]]$, meaning that $N = N'$ (as desired).
- Case $N = N_1 \mid N_2$. Since $N \stackrel{0}{\rightsquigarrow} N'$, by using Proposition 1, it holds that there exist N_1' and N_2' such that $N' = N_1' \mid N_2'$, together with $N_1 \stackrel{0}{\rightsquigarrow} N_1'$ and $N_2 \stackrel{0}{\rightsquigarrow} N_2'$. By induction the reductions $N_1 \stackrel{0}{\rightsquigarrow} N_1'$ and $N_2 \stackrel{0}{\rightsquigarrow} N_2'$ imply that $N_1' = N_1$ and

$N_2 = N_2'$, respectively. Thus $N_1' = N_1' \mid N_2' = N_1 \mid N_2$, meaning that $N = N'$ (as desired).

2. We proceed by induction on the structure of N.

- Case $N = l[[\mathbf{0}]]$. Since $N \xrightarrow{t} N'$ and $N \xrightarrow{t} N''$, by using Proposition 1, it holds that $N' = l[[\mathbf{0}]]$ and $N'' = l[[\mathbf{0}]]$, respectively, meaning that $N' = N''$ (as desired).
- Case $N = l[[a^{\Delta t'}!\langle v\rangle \text{ then } P \text{ else } Q \rhd K]]$. Since $N \xrightarrow{t} N'$ and $N \xrightarrow{t} N''$, by using Proposition 1, it holds that $N' = l[[a^{\Delta t'-t}!\langle v\rangle \text{ then } P \text{ else } Q \rhd K]]$ and $N'' = l[[a^{\Delta t'-t}!\langle v\rangle \text{ then } P \text{ else } Q \rhd K]]$, respectively, meaning that $N' = N''$ (as desired).
- Case $N = l[[a^{\Delta t'}?(u) \text{ then } P \text{ else } Q \rhd K]]$. Since $N \xrightarrow{t} N'$ and $N \xrightarrow{t} N''$, by using Proposition 1, it holds that $N' = l[[a^{\Delta t'-t}?(u) \text{ then } P \text{ else } Q \rhd K]]$ and $N'' = l[[a^{\Delta t'-t}?(u) \text{ then } P \text{ else } Q \rhd K]]$, respectively, meaning that $N' = N''$ (as desired).
- Case $N = l[[\text{go}^{t'} l' \text{ then } P \rhd K]]$. Since $N \xrightarrow{t} N'$ and $N \xrightarrow{t} N''$, by using Proposition 1, it holds that $N' = l[[\text{go}^{t'-t} l' \text{ then } P \rhd K]]$ and $N'' = l[[\text{go}^{t'-t} l' \text{ then } P \rhd K]]$, respectively, meaning that $N' = N''$ (as desired).
- Case $N = N_1 \mid N_2$. Since $N \xrightarrow{t} N'$, by using Proposition 1, it holds that there exist N_1' and N_2' such that $N' = N_1' \mid N_2'$, together with $N_1 \xrightarrow{t} N_1'$ and $N_2 \xrightarrow{t} N_2'$. Similarly, since $N \xrightarrow{t} N''$, by using Proposition 1, it holds that there exist N_1'' and N_2'' such that $N'' = N_1'' \mid N_2''$, together with $N_1 \xrightarrow{t} N_1''$ and $N_2 \xrightarrow{t} N_2''$. By induction, $N_1 \xrightarrow{t} N_1'$ and $N_1 \xrightarrow{t} N_1''$ imply that $N_1' = N_1''$, while $N_2 \xrightarrow{t} N_2'$ and $N_2 \xrightarrow{t} N_2''$ imply that $N_2' = N_2''$. Thus, $N_1' = N_1' \mid N_2' = N_1'' \mid N_2''$, meaning that $N = N'$ (as desired). □

The following theorem claims that whenever only the rules of Table 3 can be applied for two time steps of lengths t and t'', then the rules can be applied also for a time step of length $t + t'$.

Theorem 2. *If* $N \xrightarrow{t} N'' \xrightarrow{t'} N'$, *then* $N \xrightarrow{t+t'} N'$.

Proof. We proceed by induction on the structure of N.

- Case $N = l[[\mathbf{0}]]$. Since $N \xrightarrow{t} N''$ by using Proposition 1, it holds that $N'' = l[[\mathbf{0}]]$. Similarly, since $N'' \xrightarrow{t'} N'$ by using Proposition 1, it holds that $N' = l[[\mathbf{0}]]$. Rule (DSTOP) can be used for network N, namely $N \xrightarrow{t+t'} l[[\mathbf{0}]] = N'$ (as desired).
- Case $N = l[[a^{\Delta t''}!\langle v\rangle \text{ then } P \text{ else } Q \rhd K]]$. Since $N \xrightarrow{t} N''$ by using Proposition 1, it holds that $N'' = l[[a^{\Delta t''-t}!\langle v\rangle \text{ then } P \text{ else } Q \rhd K]]$, where $t'' \geq t \geq 0$. Similarly, since $N'' \xrightarrow{t'} N'$ by using Proposition 1, it holds that $N' = l[[a^{\Delta(t''-t)-t'}!\langle v\rangle \text{ then } P \text{ else } Q \rhd K]]$, where $t'' - t \geq t' \geq 0$. Due to the fact that $0 \leq t+t' \leq t''$, rule (DGET) can be used for network N, namely $N \xrightarrow{t+t'} l[[a^{\Delta t''-(t+t')}!\langle v\rangle \text{ then } P \text{ else } Q \rhd K]] = N'$ (as desired).
- Case $N = l[[a^{\Delta t''}?(u) \text{ then } P \text{ else } Q \rhd K]]$. Since $N \xrightarrow{t} N''$ by using Proposition 1, it holds that $N'' = l[[a^{\Delta t''-t}?(u) \text{ then } P \text{ else } Q \rhd K]]$, with $t'' \geq t \geq 0$. Similarly, since $N'' \xrightarrow{t'} N'$ by using Proposition 1, it holds that $N' = l[[a^{\Delta(t''-t)-t'}?(u) \text{ then } P \text{ else } Q \rhd K]]$, where $t'' - t \geq t' \geq 0$. Due to the fact that $0 \leq t+t' \leq t''$, rule (DPUT) can be used for network N, namely $N \xrightarrow{t+t'} l[[a^{\Delta t''-(t+t')}?(u) \text{ then } P \text{ else } Q \rhd K]] = N'$ (as desired).

- Case $N = l[[\mathsf{go}^{t''}\ l' \text{ then } P \triangleright K]]$. Since $N \overset{t}{\rightsquigarrow} N''$ by using Proposition 1, it holds that $N'' = l[[\mathsf{go}^{t''-t}\ l' \text{ then } P \triangleright K]]$, where $t'' \geq t \geq 0$. Similarly, since $N'' \overset{t'}{\rightsquigarrow} N'$ by using Proposition 1, it holds that $N' = l[[\mathsf{go}^{(t''-t)-t'}\ l' \text{ then } P \triangleright K]]$, where $t'' - t \geq t' \geq 0$. Due to the fact that $0 \leq t+t' \leq t''$, rule (DMOVE) can be used for network N, namely $N \overset{t+t'}{\rightsquigarrow} l[[\mathsf{go}^{t''-(t+t')}\ l' \text{ then } P \triangleright K]] = N'$ (as desired).
- Case $N = N_1 \mid N_2$. Since $N \overset{t}{\rightsquigarrow} N''$, by using Proposition 1, it holds that $N_1 \mid N_2 \not\rightarrow$ and there exist N_1'' and N_2'' such that $N'' = N_1'' \mid N_2''$, together with $N_1 \overset{t}{\rightsquigarrow} N_1''$ and $N_2 \overset{t}{\rightsquigarrow} N_2''$. Similarly, since $N'' \overset{t'}{\rightsquigarrow} N'$, by using Proposition 1, it holds that there exist N_1' and N_2' such that $N' = N_1' \mid N_2'$, together with $N_1'' \overset{t'}{\rightsquigarrow} N_1'$ and $N_2'' \overset{t'}{\rightsquigarrow} N_2'$. By induction, $N_1 \overset{t}{\rightsquigarrow} N_1''$ and $N_1'' \overset{t'}{\rightsquigarrow} N_1'$ imply that $N_1 \overset{t+t'}{\rightsquigarrow} N_1'$, while $N_2 \overset{t}{\rightsquigarrow} N_2''$ and $N_2'' \overset{t'}{\rightsquigarrow} N_2'$ imply that $N_2' \overset{t+t'}{\rightsquigarrow} N_2'$. Since $N_1 \overset{t+t'}{\rightsquigarrow} N_1'$, $N_2 \overset{t+t'}{\rightsquigarrow} N_2'$ and $N_1 \mid N_2 \not\rightarrow$, rule (DPAR) can be used for network N, namely $N \overset{t+t'}{\rightsquigarrow} N_1' \mid N_2' = N'$ (as desired). □

Regarding the knowledge of an agent, we have the following result showing that any given agent can be obtained starting from an agent without any knowledge.

Proposition 2. *If* $N'' = l[[P'' \triangleright K'']]$ *with* $K'' \neq \emptyset$, *then*
there exists $N' = l[[P' \triangleright K']]$ *with* $K' = \emptyset$ *such that* $N' \Longrightarrow^* N''$.

Proof. We proceed by induction on the structure of K''.

- Consider $K'' = \langle f \mid v; \emptyset \rangle$. According to rule (CREATE), this knowledge can be obtain from a process $P' = create(\langle f \mid v; \emptyset \rangle)$ then P''. This implies that for $N' = l[[P' \triangleright K']]$ with $K' = \emptyset$, it holds that $N' \xrightarrow{create_f@l} N''$ (as desired).
- Consider $K'' = \langle f \mid v; \emptyset \rangle\ K$, with $K \neq \emptyset$. By induction, there exists a process P able to create the knowledge K. This implies that for $N = l[[P \triangleright K]]$ with $K \neq \emptyset$, it holds that $N \Longrightarrow^* N''$. According to rule (CREATE), knowledge K'' can be obtain starting from knowledge K by using a process $P' = create(\langle f \mid v; \emptyset \rangle)$ then P. This implies that for $N' = l[[P' \triangleright K']]$ with $K' = \emptyset$, it holds that $N' \xrightarrow{create_f@l} N \Longrightarrow^* N''$ (as desired).
- Consider $K'' = \langle f' \mid v'; \ldots \langle f'' \mid v''; \langle f \mid v; \emptyset \rangle K_1 \rangle\ K_2 \rangle\ K_3$. By induction, there exists a process P able to create the knowledge $K = \langle f' \mid v'; \ldots \langle f'' \mid v''; K_1 \rangle\ K_2 \rangle\ K_3$. This implies that for $N = l[[P \triangleright K]]$ with $K \neq \emptyset$, it holds that $N \Longrightarrow^* N''$. According to rule (EXTEND), knowledge K'' can be obtain starting from knowledge K by using a process $P' = update(p/f, v)$ then P, where $p = /f' \ldots /f''$. This implies that for $N' = l[[P' \triangleright K']]$ with $K' = \emptyset$, it holds that $N' \xrightarrow{upd_p@l} N \Longrightarrow^* N''$ (as desired). □

The next result is a consequence of the previous one; it claims that any given network in knowTIMO can be obtained starting from a network containing only agents without knowledge.

Theorem 3. *If* $N'' = l_1[[P_{11}'' \triangleright K_{11}'' \mid\mid \ldots \mid\mid P_{1n}'' \triangleright K_{1n}'']] \mid \ldots \mid l_m[[P_{m1}'' \triangleright K_{m1}'' \mid\mid \ldots \mid\mid P_{mn}'' \triangleright K_{mn}'']]$, *then there exists* $N' = l_1[[P_{11}' \triangleright K_{11}' \mid\mid \ldots \mid\mid P_{1n}' \triangleright K_{1n}']] \mid \ldots \mid l_m[[P_{m1}' \triangleright K_{m1}' \mid\mid \ldots \mid\mid P_{mn}' \triangleright K_{mn}']]$ *with* $K_{ij}' = \emptyset$ $(1 \leq i \leq m, 1 \leq j \leq n)$ *such that* $N' \Longrightarrow^* N''$.

The following example illustrates how agents communicate and make use of their knowledge.

Example 1. *To illustrate how multi-agent systems can be described in* knowTIMO, *we adapt the travel agency example from [3], where all the involved agents have a cyclic behaviour. Consider a travel agency with seven offices (one central and six locals) and five employees (two executives and three travel agents). As the agency is understaffed and all local offices need to be used from time to time, the executives meet with the agents daily at the central office in order to assign them local*

offices where they sell travel packages by interacting with potential customers. We consider two customers that are willing to visit the local offices closer to their homes. In what follows we show how each of the involved agents can be described by using the knowTiMo *syntax.*

Each day, agent A_1 *executes the action* go^{10} office *in order to move after* 10 *time units from location* $\mathrm{home}_{\mathrm{A1}}$ *to the central* office. *After reaching the central office, in order to find out at which local office will work for the rest of the day, it executes the action* $b^{\Delta 5}?(\mathrm{newloc})$ *to try to communicate with any of the executives in the next* 5 *time units. The location variable* newloc *is needed to model a dynamic evolution based on the local office assigned by an available executive. After successfully communicating with an executive, the agent* A_1 *moves to location* office_i *after* 5 *time units in order to communicate with potential customers using channel* a_i *in order to sell a travel package towards location* $\mathrm{dest}_{\mathrm{A1}}$ *at the cost of* 100 *monetary units. After each working day, the agent returns home by executing the action* $\mathsf{go}^3\ \mathrm{home}_{A1}$. *The agents* A_2 *and* A_3 *behave similarly to* A_1, *except that they begin and end their days at different locations, work locally at different offices and the travel packages they advertise are different.*

Formally, the travel agents are described by the recursive processes $\mathrm{A_X}(\mathrm{home_{AX}}) \rhd K_{\mathrm{AX}}$*:*

$$\mathrm{A_X}(\mathrm{home_{AX}}) = \mathsf{go}^{10}\ \mathrm{office}\ \mathsf{then}\ \mathrm{A_X}(\mathrm{office})$$

$$\mathrm{A_X}(\mathrm{office}) = b^{\Delta 5}?(\mathrm{newloc})\ \mathsf{then}\ (\mathsf{go}^5\ \mathrm{newloc}\ \mathsf{then}\ \mathrm{A_X}(\mathrm{newloc}))\ \mathsf{else}\ \mathrm{A_X}(\mathrm{office})$$

$$\mathrm{A_X}(\mathrm{office_i}) = \mathit{update}(/\mathit{work}, \mathrm{office}_i)\ \mathsf{then}\ a_i^{\Delta 9}!\langle K_{AX}(/\mathit{work}/\mathit{dest}), K_{AX}(/\mathit{work}/\mathit{price})\rangle\ \mathsf{then}\ \mathsf{go}^3\ \mathrm{home_{AX}}\ \mathsf{then}\ \mathrm{A_X}(\mathrm{home_{AX}})\ \mathsf{else}\ \mathsf{go}^3\ \mathrm{home_{AX}}\ \mathsf{then}\ \mathrm{A_X}(\mathrm{home_{AX}})$$

$$K_{\mathrm{AX}} = \langle \mathit{work} \mid \mathrm{office}; \langle \mathit{dest} \mid \mathit{dest}_{AX}\rangle\ \langle \mathit{price} \mid 100 \cdot X\rangle\rangle.$$

The identifiers AX *(*$1 \leq X \leq 3$*) are uniquely assigned to the three travel agents, and* office_i *(*$1 \leq i \leq 6$*) indicate the six local offices.*

Given the knowledge K_{AX} *defined above, we exemplify how it can be used for some queries:*

- $K_{AX}(/\mathit{work}/\mathit{price})$ *is used to retrieve the price value* $100 \cdot X$ *by following the path* $/\mathit{work}/\mathit{price}$ *in* K_{AX}*;*
- $K_{AX}(/\mathit{work}[K_{AX}(/\mathit{work}/\mathit{price}) < 200])$ *returns the local office in which the agent is trying to sell its travel package whenever the price of the package available by following the path* $/\mathit{work}/\mathit{price}$ *is below* 200 *monetary units.*

Executives E_1 *and* E_2 *are placed in the central* office, *being available for communication on channel b for* 5 *time minutes. In this way, they can assign to the travel agents (in a cyclic manner) the locations* office_1, office_3, office_5, *and the locations* office_2, office_4, office_6, *respectively. Formally, the executives are described by* $\mathrm{E_X}(\mathrm{office_Y}) \rhd K_{\mathrm{EX}}$*:*

$$\mathrm{E_X}(\mathrm{office_Y}) = \mathit{update}(/\mathit{work}, \mathrm{office}_Y)\ \mathsf{then}\ b^{\Delta 5}!\langle K_{EX}(/\mathit{work})\rangle\ \mathsf{then}\ \mathrm{E_X}(\mathrm{office_{Y+2}})\ \mathsf{else}\ \mathrm{E_X}(\mathrm{office_Y})$$

$$K_{\mathrm{EX}} = \emptyset.$$

The identifiers EX *(with* $1 \leq X \leq 2$*) are uniquely assigned to the two executives, while* office_Y *(with* $Y \in \{X, X+2, X+4\}$*) indicate the local offices that each executive* EX *can assign to travel agents. Defining the index of the local offices in this way ensures that the executives assign the existing local offices in a cyclic way.*

The client C_1 *initially resides at location* $\mathrm{home_{C1}}$*; being interested in a travel package, client* C_1 *is willing to visit the local offices closer to his location, namely* office_1, office_2, *and* office_3. *For each of these three local offices, the visit has two possible outcomes: if client* C_1 *interacts with an agent then it will acquire a travel offer, while if the highoffice is closed then client* C_1 *moves to the next local office from its itinerary. Once its journey through the three local offices ends, client* C_1 *returns home whenever was unable to collect any travel offer, while goes at the destination for which he has to pay the lowest amount whenever got at least one offer. After the holiday period ends, client* C_1 *returns home, where can restart the process of searching for a holiday destination. Client* C_2 *behaves in a similar manner as client* C_1 *does, except looking for the most expensive*

travel package while visiting the local offices office_4, office_5 *and* office_6. *Formally, the clients are described by* $\text{C}_\text{X}(\text{home}_\text{CX}) \triangleright K_\text{CX}$:

$$\begin{aligned}
&\text{C}_\text{X}(\text{home}_\text{CX}) = \text{go}^{13}\ \text{office}_{Z+1}\ \text{then}\ \text{C}_\text{X}(\text{office}_{Z+1})\\
&\text{C}_\text{X}(\text{office}_{Z+1}) = a^{\Delta 4}_{Z+1}?(\text{dest}_{\text{CX},1}, \text{cost}_{\text{CX},1})\\
&\qquad\qquad \text{then}\ \textit{update}(/\textit{agency}[\textit{test}_{Z+1}]/\textit{dest}, \textit{dest}_{CX,1})\\
&\qquad\qquad\qquad \text{then}\ \textit{update}(/\textit{agency}[\textit{test}_{Z+1}]/\textit{price}, \textit{cost}_{CX,1})\\
&\qquad\qquad\qquad\qquad \text{then}\ \text{go}^2\ \text{office}_{Z+2}\ \text{then}\ \text{C}_\text{X}(\text{office}_{Z+2})\\
&\qquad\qquad \text{else}\ \textit{update}(/\textit{agency}[\textit{test}_{Z+1}]/\textit{dest}, \varepsilon\ )\\
&\qquad\qquad\qquad \text{then}\ \textit{update}(/\textit{agency}[\textit{test}_{Z+1}]/\textit{price}, \varepsilon\ )\\
&\qquad\qquad\qquad\qquad \text{then}\ \text{go}^2\ \text{office}_{Z+2}\ \text{then}\ \text{C}_\text{X}(\text{office}_{Z+2})\ ,\\
&\qquad\qquad \textit{where}\ \textit{test}_{Z+1} = (K_{CX}(/\textit{agency}) = \text{office}_{Z+1})\\
&\text{C}_\text{X}(\text{office}_{Z+2}) = a^{\Delta 4}_{Z+2}?(\text{dest}_{\text{CX},2}, \text{cost}_{\text{CX},2})\\
&\qquad\qquad \text{then}\ \textit{update}(/\textit{agency}[\textit{test}_{Z+2}]/\textit{dest}, \textit{dest}_{CX,2})\\
&\qquad\qquad\qquad \text{then}\ \textit{update}(/\textit{agency}[\textit{test}_{Z+2}]/\textit{price}, \textit{cost}_{CX,2})\\
&\qquad\qquad\qquad\qquad \text{then}\ \text{go}^3\ \text{office}_{Z+3}\ \text{then}\ \text{C}_\text{X}(\text{office}_{Z+3})\\
&\qquad\qquad \text{else}\ \textit{update}(/\textit{agency}[\textit{test}_{Z+2}]/\textit{dest}, \varepsilon\ )\\
&\qquad\qquad\qquad \text{then}\ \textit{update}(/\textit{agency}[\textit{test}_{Z+2}]/\textit{price}, \varepsilon\ )\\
&\qquad\qquad\qquad\qquad \text{then}\ \text{go}^2\ \text{office}_{Z+3}\ \text{then}\ \text{C}_\text{X}(\text{office}_{Z+3})\ ,\\
&\qquad\qquad \textit{where}\ \textit{test}_{Z+2} = (K_{CX}(/\textit{agency}) = \text{office}_{Z+2})\\
&\text{C}_\text{X}(\text{office}_{Z+3}) = a^{\Delta_4}_{Z+3}?(\text{dest}_{\text{CX},3}, \text{cost}_{\text{CX},3})\\
&\qquad\qquad \text{then}\ \textit{update}(/\textit{agency}[\textit{test}_{Z+3}]/\textit{dest}, \textit{dest}_{CX,3})\\
&\qquad\qquad\qquad \text{then}\ \textit{update}(/\textit{agency}[\textit{test}_{Z+3}]/\textit{price}, \textit{cost}_{CX,3})\\
&\qquad\qquad\qquad\qquad \text{then}\ \ \text{C}_\text{X}(\text{next}_\text{CX})\\
&\qquad\qquad \text{else}\ \textit{update}(/\textit{agency}[\textit{test}_{Z+3}]/\textit{dest}, \varepsilon\ )\\
&\qquad\qquad\qquad \text{then}\ \textit{update}(/\textit{agency}[\textit{test}_{Z+3}]/\textit{price}, \varepsilon\ )\\
&\qquad\qquad\qquad\qquad \text{then}\ \text{C}_\text{X}(\text{next}_\text{CX})\ ,\\
&\qquad\qquad \textit{where}\ \textit{test}_{Z+3} = (K_{CX}(/\textit{agency}) = \text{office}_{Z+3})\\
&\text{C}_\text{X}(\text{next}_\text{CX}) = \textit{if}\ \text{testX}\ \text{then}\ (\text{go}^5\ \text{next}_\text{CX}\ \text{then}\ \text{C}_\text{X}(\text{next}_\text{CX}))\\
&\qquad\qquad\qquad\qquad\qquad \text{else}\ \ (\text{go}^5\ \text{home}_\text{CX}\ \text{then}\ \text{C}_\text{X}(\text{home}_\text{CX}))\\
&\text{C}_\text{X}(\text{dest}_{\text{CX,i}}) = \text{go}^5\ \text{dest}_\text{CX}\ \text{then}\ \text{C}_\text{X}(\text{home}_\text{CX})\\
&K_\text{CX} = \langle \textit{agency} \mid \text{office}_{Z+1}; \langle \textit{dest} \mid \varepsilon\ \rangle \langle \textit{price} \mid \varepsilon\ \rangle\rangle\\
&\qquad\ \langle \textit{agency} \mid \text{office}_{Z+2}; \langle \textit{dest} \mid \varepsilon\ \rangle \langle \textit{price} \mid \varepsilon\ \rangle\rangle\\
&\qquad\ \langle \textit{agency} \mid \text{office}_{Z+3}; \langle \textit{dest} \mid \varepsilon\ \rangle \langle \textit{price} \mid \varepsilon\ \rangle\rangle\ .
\end{aligned}$$

The identifiers CX *(with* $1 \leq X \leq 2$*) are uniquely assigned to the two clients, the identifiers* $\text{dest}_{\text{CX,i}}$ *uniquely identify the possible destinations the clients* CX *can visit, while* $Z = 3 * (X - 1)$ *(with* $X \in \{1, 2\}$*) are used to identify the local offices for each of the clients.*
The tests used above are:

$$\text{testX} = \neg(K_{CX}(/\textit{agency}/\textit{price}) = \varepsilon\ ),$$

$$\text{next}_\text{CX} = \begin{cases} \text{K}_\text{CX}(/\textit{agency}[\textit{test}_{min}]/\textit{dest}_{CX,i}) \\ \qquad \textit{if}\ X = 1\ \textit{and}\ K_{CX}(/\textit{agency}/\textit{price}) = \textit{min}_{j\in\{1,2,3\}}\text{cost}_{\text{CX,j}} \in \mathbb{N}; \\ \text{K}_\text{CX}(/\textit{agency}[\textit{test}_{max}]/\textit{dest}_{CX,i}) \\ \qquad \textit{if}\ X = 2\ \textit{and}\ K_{CX}(/\textit{agency}/\textit{price}) = \textit{max}_{j\in\{1,2,3\}}\text{cost}_{\text{CX,j}} \in \mathbb{N}; \\ \text{home}_\text{CX} \qquad \textit{otherwise}. \end{cases}$$

The initial state of the system given as the knowTiMo *network N is:*

$$\begin{aligned}
&\text{home}_{\text{A1}}[[\text{A}_1(\text{home}_{\text{A1}}) \triangleright K_{\text{A1}}]] \mid \text{home}_{\text{A2}}[[\text{A}_2(\text{home}_{\text{A2}}) \triangleright K_{\text{A2}}]]\\
&\mid \text{home}_{\text{A3}}[[\text{A}_3(\text{home}_{\text{A3}}) \triangleright K_{\text{A3}}]] \mid \text{office}[[\text{E}_1(\text{office}_1) \triangleright K_{\text{E1}} \mid\mid \text{E}_2(\text{office}_2) \triangleright K_{\text{E2}}]]\\
&\mid \text{home}_{\text{C1}}[[\text{C}_1(\text{home}_{\text{C1}}) \triangleright \text{K}_{\text{C1}}] \mid \text{home}_{\text{C2}}[[\text{C}_2(\text{home}_{\text{C2}}) \triangleright \text{K}_{\text{C2}}] \mid N',
\end{aligned}$$

where N' *stands for:*

$$\begin{aligned}
&\text{office}_1[[\mathbf{0}]] \mid \text{office}_2[[\mathbf{0}]] \mid \text{office}_3[[\mathbf{0}]] \mid \text{office}_4[[\mathbf{0}]] \mid \text{office}_5[[\mathbf{0}]] \mid \text{office}_6[[\mathbf{0}]]\\
&\mid \text{dest}_1[[\mathbf{0}]] \mid \text{dest}_2[[\mathbf{0}]] \mid \text{dest}_3[[\mathbf{0}]].
\end{aligned}$$

In what follows we show how some of the rules of Tables 2 and 3 are applied such that network N evolves. Since the network N is defined by means of recursive processes, in order to execute their actions we need to use the rules (CALL) *and* (PAR) *for unfolding, namely*

$$\xrightarrow{\{call,call,call,call,call,call\}} \qquad \text{(CALL), (PAR)}$$

$$\begin{array}{l}
\mathsf{home}_{A1}[[(\mathsf{go}^{10}\ \mathsf{office}\ \mathsf{then}\ A_1(\mathsf{office}) \rhd K_{A1}]] \\
\mathsf{home}_{A2}[[(\mathsf{go}^{10}\ \mathsf{office}\ \mathsf{then}\ A_2(\mathsf{office}) \rhd K_{A2}]] \\
\mathsf{home}_{A3}[[(\mathsf{go}^{10}\ \mathsf{office}\ \mathsf{then}\ A_3(\mathsf{office}) \rhd K_{A3}]] \\
\mid \mathsf{office}[[update(/work, \mathsf{office}_1) \\
\qquad \mathsf{then}\ b^{\Delta 5}!\langle K_{E1}(/work)\rangle\ \mathsf{then}\ E_1(\mathsf{office}_3) \\
\qquad\qquad\qquad\qquad \mathsf{else}\ E_1(\mathsf{office}_1) \\
\qquad \rhd K_{E1} \\
\quad || \ update(/work, \mathsf{office}_2) \\
\qquad \mathsf{then}\ b^{\Delta 5}!\langle K_{E2}(/work)\rangle\ \mathsf{then}\ E_2(\mathsf{office}_4) \\
\qquad\qquad\qquad\qquad \mathsf{else}\ E_2(\mathsf{office}_2) \\
\qquad \rhd K_{E2}]] \\
\mid \mathsf{home}_{C1}[[\mathsf{go}^{13}\ \mathsf{office}_1\ \mathsf{then}\ C_1(\mathsf{office}_1)]] \\
\mid \mathsf{home}_{C2}[[\mathsf{go}^{13}\ \mathsf{office}_4\ \mathsf{then}\ C_2(\mathsf{office}_4)]] \\
\mid N'.
\end{array}$$

The next step is represented by the two updates performed by the executives; thus, the rules (EXTEND) *and* (PAR) *are applied several times. Since the existing knowledge of the two executives is currently* $\emptyset$*, this means that these updates extend in fact their knowledge.*

$$\xrightarrow{\{upd,upd\}} \qquad \text{(EXTEND), (PAR)}$$

$$\begin{array}{l}
\mathsf{home}_{A1}[[(\mathsf{go}^{10}\ \mathsf{office}\ \mathsf{then}\ A_1(\mathsf{office}) \rhd K_{A1}]] \\
\mathsf{home}_{A2}[[(\mathsf{go}^{10}\ \mathsf{office}\ \mathsf{then}\ A_2(\mathsf{office}) \rhd K_{A2}]] \\
\mathsf{home}_{A3}[[(\mathsf{go}^{10}\ \mathsf{office}\ \mathsf{then}\ A_3(\mathsf{office}) \rhd K_{A3}]] \\
\mid \mathsf{office}[[\ b^{\Delta 5}!\langle K_{E1}(/work)\rangle\ \mathsf{then}\ E_1(\mathsf{office}_3) \\
\qquad\qquad\qquad\qquad \mathsf{else}\ E_1(\mathsf{office}_1) \\
\qquad \rhd \langle work \mid \mathsf{office}_1; \emptyset\rangle \\
\quad ||\ \ b^{\Delta 5}!\langle K_{E2}(/work)\rangle\ \mathsf{then}\ E_2(\mathsf{office}_4) \\
\qquad\qquad\qquad\qquad \mathsf{else}\ E_2(\mathsf{office}_2) \\
\qquad \rhd \langle work \mid \mathsf{office}_2; \emptyset\rangle]] \\
\mid \mathsf{home}_{C1}[[\mathsf{go}^{13}\ \mathsf{office}_1\ \mathsf{then}\ C_1(\mathsf{office}_1)]] \\
\mid \mathsf{home}_{C2}[[\mathsf{go}^{13}\ \mathsf{office}_4\ \mathsf{then}\ C_2(\mathsf{office}_4)]] \\
\mid N'.
\end{array}$$

Since the rules of Table 2 are not applicable to the above network, then only time passing can be applied by using the rules of Table 3. The rules (DMOVE), (DGET) *and* (DPAR) *can be applied for* $t = 5$*, namely the maximum time units that can be performed.*

$$\overset{5}{\rightsquigarrow} \qquad \text{(DMOVE), (DGET), (DPAR)}$$

$$\begin{array}{l}
\mathsf{home}_{A1}[[(\mathsf{go}^{5}\ \mathsf{office}\ \mathsf{then}\ A_1(\mathsf{office}) \rhd K_{A1}]] \\
\mathsf{home}_{A2}[[(\mathsf{go}^{5}\ \mathsf{office}\ \mathsf{then}\ A_2(\mathsf{office}) \rhd K_{A2}]] \\
\mathsf{home}_{A3}[[(\mathsf{go}^{5}\ \mathsf{office}\ \mathsf{then}\ A_3(\mathsf{office}) \rhd K_{A3}]] \\
\mid \mathsf{office}[[\ b^{\Delta 0}!\langle K_{E1}(/work)\rangle\ \mathsf{then}\ E_1(\mathsf{office}_3) \\
\qquad\qquad\qquad\qquad \mathsf{else}\ E_1(\mathsf{office}_1) \\
\qquad \rhd \langle work \mid \mathsf{office}_1; \emptyset\rangle \\
\quad ||\ \ b^{\Delta 0}!\langle K_{E2}(/work)\rangle\ \mathsf{then}\ E_2(\mathsf{office}_4) \\
\qquad\qquad\qquad\qquad \mathsf{else}\ E_2(\mathsf{office}_2) \\
\qquad \rhd \langle work \mid \mathsf{office}_2; \emptyset\rangle]] \\
\mid \mathsf{home}_{C1}[[\mathsf{go}^{8}\ \mathsf{office}_1\ \mathsf{then}\ C_1(\mathsf{office}_1)]] \\
\mid \mathsf{home}_{C2}[[\mathsf{go}^{8}\ \mathsf{office}_4\ \mathsf{then}\ C_2(\mathsf{office}_4)]] \\
\mid N'.
\end{array}$$

Since after 5 *time units of the evolution there are no agents to communicate with the executives on channel b, then the rules* (PUT0) *and* (PAR) *are applied such that the* else *branches of the two executives are chosen to be executed next.*

$$\xrightarrow{\{b!^{\Delta 0}@office,\ b!^{\Delta 0}@office\}} \qquad \text{(PUT0), (PAR)}$$

$$\begin{array}{l}
\mathrm{home}_{A1}[[(\mathrm{go}^5\ \mathrm{office}\ \mathrm{then}\ A_1(\mathrm{office}) \rhd K_{A1}]] \\
\mathrm{home}_{A2}[[(\mathrm{go}^5\ \mathrm{office}\ \mathrm{then}\ A_2(\mathrm{office}) \rhd K_{A2}]] \\
\mathrm{home}_{A3}[[(\mathrm{go}^5\ \mathrm{office}\ \mathrm{then}\ A_3(\mathrm{office}) \rhd K_{A3}]] \\
\mid \mathrm{office}[[\ E_1(\mathrm{office}_1) \rhd \langle work \mid \mathrm{office}_1; \emptyset\rangle \\
\qquad\qquad \mid\mid\ E_2(\mathrm{office}_2) \rhd \langle work \mid \mathrm{office}_2; \emptyset\rangle]] \\
\mid \mathrm{home}_{C1}[[\mathrm{go}^8\ \mathrm{office}_1\ \mathrm{then}\ C_1(\mathrm{office}_1)]] \\
\mid \mathrm{home}_{C2}[[\mathrm{go}^8\ \mathrm{office}_4\ \mathrm{then}\ C_2(\mathrm{office}_4)]] \\
\mid N'.
\end{array}$$

Note that the evolution was deterministic during the first 5 *time units. However, since there are two executives and three travel agents into the system, the communication on channel b will take place in a nondeterministic manner, and thus there exists several possible future evolutions of the system.*

3. Behavioural Equivalences in knowTIMO

In what follows, we define and study bisimulations for multi-agent systems that consider knowledge dynamics as well as explicit time constraints for communication and migration. Since a bisimilarity is the union of all bisimulations of the same type, in order to demonstrate that two knowTIMO networks N_1 and N_2 are bisimilar it is enough to discover a bisimulation relation containing the pair (N_1, N_2). This standard bisimulation proof method is interesting for the following reasons:

- check-ups are local (only immediate transitions are used);
- No hierarchy exists between the pairs of a bisimulation, and thus we can effectively use bisimilarity to reason about infinite behaviours; this makes it different from inductive techniques, where we can reason about finite behaviour due to the required hierarchy.

3.1. Strong Timed Equivalences

Inspired by the approach taken in [4], we extend the standard notion of strong bisimilarity by allowing also timed transitions to be taken into account.

Definition 1 (Strong timed bisimulation)**.**
Let $\mathcal{R} \subseteq \mathcal{N} \times \mathcal{N}$ be a symmetric binary relation over knowTIMO *networks.*

1. *$\mathcal{R}$ is a* strong timed bisimulation *if*
   - *$(N_1, N_2) \in \mathcal{R}$ and $N_1 \xrightarrow{\lambda} N_1'$ implies that there exists $N_2' \in \mathcal{N}$ such that $N_2 \xrightarrow{\lambda} N_2'$ and $(N_1', N_2') \in \mathcal{R}$;*
   - *$(N_1, N_2) \in \mathcal{R}$ and $N_1 \overset{t}{\rightsquigarrow} N_1'$ implies that there exists $N_2' \in \mathcal{N}$ such that $N_2 \overset{t}{\rightsquigarrow} N_2'$ and $(N_1', N_2') \in \mathcal{R}$.*
2. *The* strong timed bisimilarity *is the union $\sim$ of all strong timed bisimulations $\mathcal{R}$.*

Definition 1 treats in a similar manner the timed transitions and the labelled transitions, and so the bisimilarity notion is similar to the bisimilarity notion originally given for labelled transition systems. We can prove that the relation $\sim$ is the largest strong timed bisimulation, and also an equivalence relation.

Proposition 3.

1. *Identity, inverse, composition and union of strong timed bisimulations are strong timed bisimulations.*
2. *$\sim$ is the largest strong timed bisimulation.*
3. *$\sim$ is an equivalence.*

Proof.

1. We treat each relations separately showing that it respects the conditions from Definition 1 for being a strong timed bisimulation.
   (a) The identity relation $Id_{\mathcal{R}}$ is a strong timed bisimulation.
      i. Assume $(N, N) \in Id_{\mathcal{R}}$. Consider $N \xrightarrow{\lambda} N'$; then $(N', N') \in Id_{\mathcal{R}}$.
      ii. Assume $(N, N) \in Id_{\mathcal{R}}$. Consider $N \xrightarrow{t} N'$; then $(N', N') \in Id_{\mathcal{R}}$.
   (b) The inverse of a strong timed bisimulation is a strong timed bisimulation.
      i. Assume $(N_1, N_2) \in \mathcal{R}^{-1}$, namely $(N_2, N_1) \in \mathcal{R}$. Consider $N_2 \xrightarrow{\lambda} N_2'$; then for some N_1' we have $N_1 \xrightarrow{\lambda} N_1'$ and $(N_2', N_1') \in \mathcal{R}$, namely $(N_1', N_2') \in \mathcal{R}^{-1}$. By similar reasoning, if $N_1 \xrightarrow{\lambda} N_1'$ then we can find N_2' such that $N_2 \xrightarrow{\lambda} N_2'$ and $(N_1', N_2') \in \mathcal{R}^{-1}$.
      ii. Assume $(N_1, N_2) \in \mathcal{R}^{-1}$, namely $(N_2, N_1) \in \mathcal{R}$. Consider $N_2 \xrightarrow{t} N_2'$; then for some N_1' we have $N_1 \xrightarrow{t} N_1'$ and $(N_2', N_1') \in \mathcal{R}$, namely $(N_1', N_2') \in \mathcal{R}^{-1}$. By similar reasoning, if $N_1 \xrightarrow{t} N_1'$ then we can find N_2' such that $N_2 \xrightarrow{t} N_2'$ and $(N_1', N_2') \in \mathcal{R}^{-1}$.
   (c) The composition of strong timed bisimulations is a strong timed bisimulation.
      i. Assume $(N_1, N_2) \in \mathcal{R}_1\mathcal{R}_2$. Then for some N we have $(N_1, N) \in \mathcal{R}_1$ and $(N, N_2) \in \mathcal{R}_2$. Consider $N_1 \xrightarrow{\lambda} N_1'$; then for some N', since $(N_1, N) \in \mathcal{R}_1$, we have $N \xrightarrow{\lambda} N'$ and $(N_1', N') \in \mathcal{R}_1$. Also, since $(N, N_2) \in \mathcal{R}_2$ we have for some N_2' that $N_2 \xrightarrow{\lambda} N_2'$ and $(N', N_2') \in \mathcal{R}_2$. Thus, $(N_1', N_2') \in \mathcal{R}_1\mathcal{R}_2$. By similar reasoning, if $N_2 \xrightarrow{\lambda} N_2'$ then we can find N_1' such that $N_1 \xrightarrow{\lambda} N_1'$ and $(N', N_2') \in \mathcal{R}_2$.
      ii. Assume $(N_1, N_2) \in \mathcal{R}_1\mathcal{R}_2$. Then for some N we have $(N_1, N) \in \mathcal{R}_1$ and $(N, N_2) \in \mathcal{R}_2$. Consider $N_1 \xrightarrow{t} N_1'$; then for some N', since $(N_1, N) \in \mathcal{R}_1$, we have $N \xrightarrow{t} N'$ and $(N_1', N') \in \mathcal{R}_1$. Also, since $(N, N_2) \in \mathcal{R}_2$ we have for some N_2' that $N_2 \xrightarrow{t} N_2'$ and $(N', N_2') \in \mathcal{R}_2$. Thus, $(N_1', N_2') \in \mathcal{R}_1\mathcal{R}_2$. By similar reasoning, if $N_2 \xrightarrow{t} N_2'$ then we can find N_1' such that $N_1 \xrightarrow{t} N_1'$ and $(N', N_2') \in \mathcal{R}_2$.
   (d) The union of strong timed bisimulations is a strong timed bisimulation.
      i. Assume $(N_1, N_2) \in \bigcup_{i \in I} \mathcal{R}_i$. Then for some $i \in I$ we have $(N_1, N_2) \in \mathcal{R}_i$. Consider $N_1 \xrightarrow{\lambda} N_1'$; then for some N_2', since $(N_1, N_2) \in \mathcal{R}_i$, we have $N_2 \xrightarrow{\lambda} N_2'$ and $(N_1', N_2') \in \mathcal{R}_i$. Thus, $(N_1', N_2') \in \bigcup_{i \in I} \mathcal{R}_i$. By similar reasoning, if $N_2 \xrightarrow{\lambda} N_2'$ then we can find N_1' such that $N_1 \xrightarrow{\lambda} N_1'$ and $(N_1', N_2') \in \mathcal{R}_i$, namely $(N_1', N_2') \in \bigcup_{i \in I} \mathcal{R}_i$.
      ii. Assume $(N_1, N_2) \in \bigcup_{i \in I} \mathcal{R}_i$. Then for some $i \in I$ we have $(N_1, N_2) \in \mathcal{R}_i$. Consider $N_1 \xrightarrow{t} N_1'$; then for some N_2', since $(N_1, N_2) \in \mathcal{R}_i$, we have $N_2 \xrightarrow{t} N_2'$ and $(N_1', N_2') \in \mathcal{R}_i$. Thus, $(N_1', N_2') \in \bigcup_{i \in I} \mathcal{R}_i$. By similar reasoning, if $N_2 \xrightarrow{t} N_2'$ then we can find N_1' such that $N_1 \xrightarrow{t} N_1'$ and $(N_1', N_2') \in \mathcal{R}_i$, namely $(N_1', N_2') \in \bigcup_{i \in I} \mathcal{R}_i$.

2. By the previous case (the union part), $\sim$ is a strong timed bisimulation and includes any other strong timed bisimulation.
3. Proving that relation $\sim$ is an equivalence requires proving that it satisfies reflexivity, symmetry and transitivity. We consider each of them in the following:
   (a) Reflexivity: For any network N, $N \sim N$ results from the fact that the identity relation is a strong timed bisimulation.
   (b) Symmetry: If $N \sim N'$, then $(N, N') \in \mathcal{R}$ for some strong timed bisimulation $\mathcal{R}$. Hence $(N', N) \in \mathcal{R}^{-1}$, and so $N' \sim N$ because the inverse relation is a strong timed bisimulation.
   (c) Transitivity: If $N \sim N'$ and $N' \sim N''$ then $(N, N') \in \mathcal{R}_1$ and $(N', N'') \in \mathcal{R}_2$ for some strong timed bisimulations $\mathcal{R}_1$ and $\mathcal{R}_2$. Thus, $(N, N'') \in \mathcal{R}_1\mathcal{R}_2$, and so $N \sim N''$ due to the fact that the composition relation is a strong timed bisimulation. □

The next result claims that the strong timed equivalence $\sim$ among processes is preserved even if the local knowledge of the agents is expanded. This is consistent with the fact that the processes affect the same portion of their knowledge. To simplify the presentation, in what follows we assume the notations $|_{i=1}^{n}\ N_i = N_1 \mid \ldots \mid N_n$ and $||_{i=1}^{n}\ A_i = A_1 \,||\, \ldots \,||\, A_n$.

Proposition 4. *If $K'_{ij} \subseteq K''_{ij}$ for $1 \le i \le n, 1 \le j \le m$, then*
$$|_{i=1}^{n}\ l_i[[||_{j=1}^{m}\ P_{ij} \rhd K'_{ij}]] \ \sim\ |_{i=1}^{n}\ l_i[[||_{j=1}^{m}\ P_{ij} \rhd K''_{ij}]].$$

Proof. We show that $\mathcal{S}$ is a strong timed bisimulation, where:
$\mathcal{S} = \{(|_{i=1}^{n}\ l_i[[||_{j=1}^{m}\ P_{ij} \rhd K'_{ij}]]\ ,\ |_{i=1}^{n}\ l_i[[||_{j=1}^{m}\ P_{ij} \rhd K''_{ij}]])\ :\ K'_{ij} \subseteq K''_{ij}, 1 \le i \le n, 1 \le j \le m\}$.
The proof is by induction on the last performed step:

- Let us assume that $|_{i=1}^{n}\ l_i[[||_{j=1}^{m}\ P_{ij} \rhd K'_{ij}]] \xrightarrow{\lambda} N'$. Depending on the value of λ, there are several cases:
  - Consider $\lambda = a!?@l_1$. Then there exists $P_{11} = a^{\Delta t_1}!\langle v\rangle$ then P'_{11} else P''_{11} and $P_{12} = a^{\Delta t_2}?(u)$ then P'_{12} else P''_{12} such that $l_1[[P_{11} \rhd K'_{11} \,||\, P_{12} \rhd K'_{12} \,||_{j=3}^{m}\ P_{1j} \rhd K'_{1j}]]\ |_{i=2}^{n}\ l_i[[||_{j=1}^{m}\ P_{ij} \rhd K'_{ij}]] \xrightarrow{a!?@l_1} l_1[[P'_{11} \rhd K'_{11} \,||\, P'_{12} \rhd K'_{12} \,||_{j=3}^{m}\ P_{1j} \rhd K'_{1j}]]\ |_{i=2}^{n}\ l_i[[||_{j=1}^{m}\ P_{ij} \rhd K'_{ij}]] = N'$. Then there exists $N'' = [[P'_{11} \rhd K''_{11} \,||\, P'_{12} \rhd K''_{12} \,||_{j=3}^{m}\ P_{1j} \rhd K''_{1j}]]\ |_{i=2}^{n}\ l_i[[||_{j=1}^{m}\ P_{ij} \rhd K''_{ij}]]$ such that $|_{i=1}^{n}\ l_i[[||_{j=1}^{m}\ P_{ij} \rhd K''_{ij}]] \xrightarrow{a!?@l_1} N''$. Since $K'_{ij} \subseteq K''_{ij}, 1 \le i \le n, 1 \le j \le m$, clearly $(N', N'') \in \mathcal{S}$.
  - Consider $\lambda = a!^{\Delta 0}@l_1$. Then there exists $P_{11} = a^{\Delta 0}!\langle v\rangle$ then P'_{11} else P''_{11} such that $l_1[[P_{11} \rhd K'_{11} \,||_{j=2}^{m}\ P_{1j} \rhd K'_{1j}]]\ |_{i=2}^{n}\ l_i[[||_{j=1}^{m}\ P_{ij} \rhd K'_{ij}]] \xrightarrow{a!^{\Delta 0}@l_1} l_1[[P'_{11} \rhd K'_{11} \,||_{j=2}^{m}\ P_{1j} \rhd K'_{1j}]]\ |_{i=2}^{n}\ l_i[[||_{j=1}^{m}\ P_{ij} \rhd K'_{ij}]] = N'$. Then there exists $N'' = l_1[[P'_{11} \rhd K''_{11} \,||_{j=2}^{m}\ P_{1j} \rhd K''_{1j}]]\ |_{i=2}^{n}\ l_i[[||_{j=1}^{m}\ P_{ij} \rhd K''_{ij}]]$ such that $|_{i=1}^{n}\ l_i[[||_{j=1}^{m}\ P_{ij} \rhd K''_{ij}]]) \xrightarrow{a!^{\Delta 0}@l_1} N''$. Since $K'_{ij} \subseteq K''_{ij}, 1 \le i \le n, 1 \le j \le m$, clearly $(N', N'') \in \mathcal{S}$.
  - Consider $\lambda = a?^{\Delta 0}@l_1$. Then there exists $P_{11} = a^{\Delta 0}?(u)$ then P'_{11} else P''_{11} such that $l_1[[P_{11} \rhd K'_{11} \,||_{j=2}^{m}\ P_{1j} \rhd K'_{1j}]]\ |_{i=2}^{n}\ l_i[[||_{j=1}^{m}\ P_{ij} \rhd K'_{ij}]] \xrightarrow{a?^{\Delta 0}@l_1} l_1[[P'_{11} \rhd K'_{11} \,||_{j=2}^{m}\ P_{1j} \rhd K'_{1j}]]\ |_{i=2}^{n}\ l_i[[||_{j=1}^{m}\ P_{ij} \rhd K'_{ij}]] = N'$. Then there exists $N'' = l_1[[P'_{11} \rhd K''_{11} \,||_{j=2}^{m}\ P_{1j} \rhd K''_{1j}]]\ |_{i=2}^{n}\ l_i[[||_{j=1}^{m}\ P_{ij} \rhd K''_{ij}]]$ such that $|_{i=1}^{n}\ l_i[[||_{j=1}^{m}\ P_{ij} \rhd K''_{ij}]]) \xrightarrow{a?^{\Delta 0}@l_1} N''$. Since $K'_{ij} \subseteq K''_{ij}, 1 \le i \le n, 1 \le j \le m$, clearly $(N', N'') \in \mathcal{S}$.
  - Consider $\lambda = l_1 \triangleright l_2$. Then there exists $P_{11} = \mathsf{go}^0\ l_2$ then P'_{11} such that $l_1[[P_{11} \rhd K'_{11} \,||_{j=2}^{m}\ P_{1j} \rhd K'_{1j}]]\ |_{i=2}^{n}\ l_i[[||_{j=1}^{m}\ P_{ij} \rhd K'_{ij}]] \xrightarrow{l_1 \triangleright l_2} l_1[[||_{j=2}^{m}\ P_{1j} \rhd K'_{1j}]] \mid l_2[[P'_{11} \rhd K'_{11} \,||_{j=1}^{m}\ P_{2j} \rhd K'_{2j}]]\ |_{i=3}^{n}\ l_i[[||_{j=1}^{m}\ P_{ij} \rhd K'_{ij}]] = N'$. Then there exists $N'' = l_1[[||_{j=2}^{m}\ P_{1j} \rhd K''_{1j}]] \mid l_2[[P'_{11} \rhd K''_{11} \,||_{j=1}^{m}\ P_{2j} \rhd K''_{2j}]]\ |_{i=3}^{n}\ l_i[[||_{j=1}^{m}\ P_{ij} \rhd K''_{ij}]]$ such that

$|_{i=1}^{n}\ l_i[[|]_{j=1}^{m}\ P_{ij} \rhd K''_{ij}]]) \xrightarrow{l_1 \rhd l_2} N''$. Since $K'_{ij} \subseteq K''_{ij}, 1 \leq i \leq n, 1 \leq j \leq m$, clearly $(N', N'') \in \mathcal{S}$.

- Consider $\lambda = true@l_1$. Then there exists $P_{11} = \text{if } test \text{ then } P'_{11} \text{ else } P''_{11}$, where $test@K'_{11} = true$, such that $l_1[[P_{11} \rhd K'_{11}\ ||_{j=2}^{m}\ P_{1j} \rhd K'_{1j}]]\ |_{i=2}^{n}\ l_i[[|]_{j=1}^{m}\ P_{ij} \rhd K'_{ij}]] \xrightarrow{true@l_1} l_1[[P'_{11} \rhd K'_{11}\ ||_{j=2}^{m}\ P_{1j} \rhd K'_{1j}]]\ |_{i=2}^{n}\ l_i[[|]_{j=1}^{m}\ P_{ij} \rhd K'_{ij}]] = N'$. Then there exists $N'' = l_1[[P'_{11} \rhd K''_{11}\ ||_{j=2}^{m}\ P_{1j} \rhd K''_{1j}]]\ |_{i=2}^{n}\ l_i[[|]_{j=1}^{m}\ P_{ij} \rhd K''_{ij}]]$ such that $|_{i=1}^{n}\ l_i[[|]_{j=1}^{m}\ P_{ij} \rhd K''_{ij}]]) \xrightarrow{true@l_1} N''$. Since $K'_{ij} \subseteq K''_{ij}, 1 \leq i \leq n, 1 \leq j \leq m$, clearly $(N', N'') \in \mathcal{S}$.
- Consider $\lambda = false@l_1$. Then there exists $P_{11} = \text{if } test \text{ then } P'_{11} \text{ else } P''_{11}$, where $test@K'_{11} = false$, such that $l_1[[P_{11} \rhd K'_{11}\ ||_{j=2}^{m}\ P_{1j} \rhd K'_{1j}]]\ |_{i=2}^{n}\ l_i[[|]_{j=1}^{m}\ P_{ij} \rhd K'_{ij}]] \xrightarrow{false@l_1} l_1[[P''_{11} \rhd K'_{11}\ ||_{j=2}^{m}\ P_{1j} \rhd K'_{1j}]]\ |_{i=2}^{n}\ l_i[[|]_{j=1}^{m}\ P_{ij} \rhd K'_{ij}]] = N'$. Then there exists $N'' = l_1[[P''_{11} \rhd K''_{11}\ ||_{j=2}^{m}\ P_{1j} \rhd K''_{1j}]]\ |_{i=2}^{n}\ l_i[[|]_{j=1}^{m}\ P_{ij} \rhd K''_{ij}]]$ such that $|_{i=1}^{n}\ l_i[[|]_{j=1}^{m}\ P_{ij} \rhd K''_{ij}]]) \xrightarrow{false@l_1} N''$. Since $K'_{ij} \subseteq K''_{ij}, 1 \leq i \leq n, 1 \leq j \leq m$, clearly $(N', N'') \in \mathcal{S}$.
- Consider $\lambda = create_f@l_1$. Then there exists $P_{11} = create(\langle f \mid v; \emptyset \rangle) \text{ then } P'_{11}$ such that $l_1[[P_{11} \rhd K'_{11}\ ||_{j=2}^{m}\ P_{1j} \rhd K'_{1j}]]\ |_{i=2}^{n}\ l_i[[|]_{j=1}^{m}\ P_{ij} \rhd K'_{ij}]] \xrightarrow{create_f@l_1} l_1[[P'_{11} \rhd K'_{11}\ \langle f \mid v; \emptyset \rangle\ ||_{j=2}^{m}\ P_{1j} \rhd K'_{1j}]]\ |_{i=2}^{n}\ l_i[[|]_{j=1}^{m}\ P_{ij} \rhd K'_{ij}]] = N'$. Then there exists $N'' = l_1[[P'_{11} \rhd K''_{11}\ \langle f \mid v; \emptyset \rangle\ ||_{j=2}^{m}\ P_{1j} \rhd K''_{1j}]]\ |_{i=2}^{n}\ l_i[[|]_{j=1}^{m}\ P_{ij} \rhd K''_{ij}]]$ such that $|_{i=1}^{n}\ l_i[[|]_{j=1}^{m}\ P_{ij} \rhd K''_{ij}]]) \xrightarrow{create_f@l_1} N''$. Since $K'_{ij} \subseteq K''_{ij}, 1 \leq i \leq n, 1 \leq j \leq m$, then also $K'_{11}\ \langle f \mid v; \emptyset \rangle \subseteq K''_{11}\ \langle f \mid v; \emptyset \rangle$, and clearly $(N', N'') \in \mathcal{S}$.
- Consider $\lambda = upd_p@l_1$. Then there exists $P_{11} = update(p/f, v) \text{ then } P'_{11}$ such that $l_1[[P_{11} \rhd K'_{11}\ ||_{j=2}^{m}\ P_{1j} \rhd K'_{1j}]]\ |_{i=2}^{n}\ l_i[[|]_{j=1}^{m}\ P_{ij} \rhd K'_{ij}]] \xrightarrow{upd_p@l_1} l_1[[P'_{11} \rhd K^{u'}_{11}\ ||_{j=2}^{m}\ P_{1j} \rhd K'_{1j}]]\ |_{i=2}^{n}\ l_i[[|]_{j=1}^{m}\ P_{ij} \rhd K'_{ij}]] = N'$. Then there exists $N'' = l_1[[P'_{11} \rhd K^{u''}_{11}\ ||_{j=2}^{m}\ P_{1j} \rhd K'_{1j}]]\ |_{i=2}^{n}\ l_i[[|]_{j=1}^{m}\ P_{ij} \rhd K'_{ij}]]$ such that $|_{i=1}^{n}\ l_i[[|]_{j=1}^{m}\ P_{ij} \rhd K''_{ij}]]) \xrightarrow{upd_p@l_1} N''$. Since $K'_{ij} \subseteq K''_{ij}, 1 \leq i \leq n, 1 \leq j \leq m$, then also $K^{u'}_{11} \subseteq K^{u''}_{11}$, and clearly $(N', N'') \in \mathcal{S}$.

- Let us assume that $|_{i=1}^{n}\ l_i[[|]_{j=1}^{m}\ P_{ij} \rhd K'_{ij}]] \overset{t}{\rightsquigarrow} N'$. Then there exists P'_{ij}, $1 \leq i \leq n$, $1 \leq j \leq m$, such that $|_{i=1}^{n}\ l_i[[|]_{j=1}^{m}\ P_{ij} \rhd K'_{ij}]] \overset{t}{\rightsquigarrow} |_{i=1}^{n}\ l_i[[|]_{j=1}^{m}\ P'_{ij} \rhd K'_{ij}]] = N'$. Then there exists $N'' = |_{i=1}^{n}\ l_i[[|]_{j=1}^{m}\ P'_{ij} \rhd K''_{ij}]]$ such that $|_{i=1}^{n}\ l_i[[|]_{j=1}^{m}\ P_{ij} \rhd K''_{ij}]] \overset{t}{\rightsquigarrow} N''$. Since $K'_{ij} \subseteq K''_{ij}$, $1 \leq i \leq n$, $1 \leq j \leq m$, clearly $(N', N'') \in \mathcal{S}$.

  The symmetric cases follow by similar arguments. □

The following result shows that strong timed bisimulation is preserved even after complete computational steps of two knowTiMo networks.

Proposition 5. *Let N_1, N_2 be two* knowTiMo *networks.*

If $N_1 \sim N_2$ and $N_1 \overset{\Lambda,t}{\Longrightarrow} N'_1$, then there exists $N'_2 \in \mathcal{N}$ such that $N_2 \overset{\Lambda,t}{\Longrightarrow} N'_2$ and $N'_1 \sim N'_2$.

Proof. Assuming that the finite multiset of actions Λ contains the labels $\{\lambda_1, \ldots, \lambda_k\}$, then the complete computational step $N_1 \overset{\Lambda,t}{\Longrightarrow} N'_1$ can be detailed as $N_1 \xrightarrow{\lambda_1} N^1_1 \ldots N^{k-1}_1 \xrightarrow{\lambda_k} N^k_1 \overset{t}{\rightsquigarrow} N'_1$. Since $N_1 \xrightarrow{\lambda_1} N^1_1$ and $N_1 \sim N_2$, then according to Definition 1 there exists $N^1_2 \in \mathcal{N}$ such that $N_2 \xrightarrow{\lambda_1} N^1_2$ and $N^1_1 \sim N^1_2$. The same reasoning can be applied for another k steps, meaning that there exist $N^2_2, \ldots, N^k_2, N'_2 \in \mathcal{N}$ such that $N_2 \xrightarrow{\lambda_1} N^1_2 \ldots N^{k-1}_2 \xrightarrow{\lambda_k} N^k_2 \overset{t}{\rightsquigarrow} N'_2$ and $N'_1 \sim N'_2$. By the definition of a complete computational step, it holds that $N_2 \xrightarrow{\lambda_1}$

$N_2^1 \dots N_2^{k-1} \xrightarrow{\lambda_k} N_2^k \overset{t}{\rightsquigarrow} N_2'$ can be written as $N_2 \overset{\Lambda,t}{\Longrightarrow} N_2'$. Thus, we obtained that there exists $N_2' \in \mathcal{N}$ such that $N_2 \overset{\Lambda,t}{\Longrightarrow} N_2'$ and $N_1' \sim N_2'$ (as desired). □

The next example illustrates that the relation $\sim$ is able to distinguish between agents with different knowledge if *update* operations are performed.

Example 2. *Consider that client C_2 is at location* office$_4$*, ready to communicate on channel a_4. To simplify the presentation, we take only a simplified definition of C_2 as follows:*

$$\begin{aligned} C_2'(\text{office}_4) = a_4^{\Delta 4}?(&\text{dest}_{C2,1}, \text{cost}_{C2,1}) \\ &\text{then } update(/agency[test_4]/dest, dest_{C2,1}) \\ &\text{else } update(/agency[test_4]/dest, \varepsilon\ ). \end{aligned}$$

Consider the following three networks in knowTiMo*:*

$N_1 = \text{office}_4[[C_2'(\text{office}_4) \triangleright K_{C2}]]$,
$N_1' = \text{office}_4[[C_2'(\text{office}_4) \triangleright K_{C2}']]$,
$N_1'' = \text{office}_4[[C_2'(\text{office}_4) \triangleright K_{C2}'']]$,

where the knowledge of the agents is defined as:

$K_{C2} = \langle agency \mid \text{office}_4; \langle dest \mid \varepsilon \rangle \langle price \mid \varepsilon \rangle \rangle$,
$K_{C2}' = \langle agency \mid \text{office}_5; \langle dest \mid \varepsilon \rangle \langle price \mid \varepsilon \rangle \rangle$,
$K_{C2}'' = \emptyset$.

According to Definition 1, it holds that $N_1' \sim N_1''$, while $N_1 \not\sim N_1'$ and $N_1 \not\sim N_1''$. This is due to the fact that while all three networks are able to perform a time step of length 4 *and to choose the* else *branch, only network N_1 is able to perform the update operation. Formally:*

$$N_1 \overset{4}{\rightsquigarrow} N_2 \xrightarrow{false@\text{office}_4} N_3 \xrightarrow{upd_{/agency[test_4]/dest}@\text{office}_4} N_4$$

and

$$N_1' \overset{4}{\rightsquigarrow} N_2' \xrightarrow{false@\text{office}_4} N_3' \not\xrightarrow{upd_{/agency[test_4]/dest}@\text{office}_4}$$
$$N_1'' \overset{4}{\rightsquigarrow} N_2'' \xrightarrow{false@\text{office}_4} N_3'' \not\xrightarrow{upd_{/agency[test_4]/dest}@\text{office}_4},$$

where the networks N_2, N_3, N_4, N_2', N_3', N_2'' and N_3'' in knowTiMo *are obtained by using the rules of Tables 2 and 3.*

3.2. Strong Bounded Timed Equivalences

We provide some notations used in the rest of the paper:

- A *timed relation* over the set $\mathcal{N}$ of networks is any relation $\mathcal{R} \subseteq \mathcal{N} \times \mathbb{N} \times \mathcal{N}$.
- The *identity timed relation* is
$$\iota \overset{df}{=} \{(N, t, N) \mid N \in \mathcal{N},\ t \in \mathbb{N}\}.$$
- The *inverse of a timed relation* $\mathcal{R}$ is
$$\mathcal{R}^{-1} \overset{df}{=} \{(N_2, t, N_1) \mid (N_1, t, N_2) \in \mathcal{R}\}.$$
- The *composition of timed relations* $\mathcal{R}_1$ and $\mathcal{R}_2$ is
$$\mathcal{R}_1\mathcal{R}_2 \overset{df}{=} \{(N, t, N'') \mid \exists N' \in \mathcal{N} : (N, t, N') \in \mathcal{R}_1 \wedge (N', t, N'') \in \mathcal{R}_2\}\ .$$
- If $\mathcal{R}$ is a timed relation and $t \in \mathbb{N}$, then
$$\mathcal{R}_t \overset{df}{=} \{(N_1, N_2) \mid (N_1, t, N_2) \in \mathcal{R}\}$$
is $\mathcal{R}$'s *t-projection*. We also denote $\mathcal{R}_\infty \overset{df}{=} \bigcup_{t \in \mathbb{N}} \mathcal{R}_t$.
- A timed relation $\mathcal{R}$ is a timed equivalence if $\mathcal{R}_\infty$ is an equivalence relation, and is an equivalence up-to time $t \in \mathbb{N}$ if $\bigcup_{0 \le t' < t} \mathcal{R}_{t'}$ is an equivalence relation.

The equivalence $\sim$ requires an exact match of transitions of two networks during their entire evolutions. Sometimes this requirement is too strong. In many situations this requirement is relaxed [5], and real-time systems are allowed to behave in an expected way up to a certain amount t of time units. This impels one to define *bounded* timed equivalences up-to a given time t.

Definition 2 (Strong bounded timed bisimulation)**.**
Let $\mathcal{R} \subseteq \mathcal{N} \times \mathbb{N} \times \mathcal{N}$ be a symmetric timed relation on $\mathbb{N}$ and on networks in knowTiMo*.*

1. *$\mathcal{R}$ is a* strong bounded timed bisimulation *if*
   - *$(N_1, t, N_2) \in \mathcal{R}$ and $N_1 \xrightarrow{\lambda} N_1'$ implies that there exists $N_2' \in \mathcal{N}$ such that $N_2 \xrightarrow{\lambda} N_2'$ and $(N_1', t, N_2') \in \mathcal{R}$;*
   - *$(N_1, t, N_2) \in \mathcal{R}$ and $N_1 \overset{t'}{\rightsquigarrow} N_1'$ implies that there exists $N_2' \in \mathcal{N}$ such that $N_2 \overset{t'}{\rightsquigarrow} N_2'$ and $(N_1', t - t', N_2') \in \mathcal{R}$.*
2. *The* strong bounded timed bisimilarity *is the union $\simeq$ of all strong bounded timed bisimulations $\mathcal{R}$.*

The following results illustrate some properties of the strong bounded timed bisimulations. In particular, we prove that the equivalence relation $\simeq$ (that is strictly included in relation $\sim$) is the largest strong bounded timed bisimulation.

Proposition 6.
1. *Identity, inverse, composition and union of strong bounded timed bisimulations are strong bounded timed bisimulations.*
2. *$\simeq$ is the largest strong bounded timed bisimulation.*
3. *$\simeq$ is a timed equivalence.*
4. *$\simeq \subsetneq \sim$.*

Proof.
1. We treat each relations separately showing that it respects the conditions from Definition 2 for being a strong bounded timed bisimulation.
   (a) The identity relation ι is a strong bounded timed bisimulation.
      i. Assume $(N, t, N) \in \iota$. Consider $N \xrightarrow{\lambda} N'$; then $(N', t, N') \in \iota$.
      ii. Assume $(N, t, N) \in \iota$. Consider $N \overset{t'}{\rightsquigarrow} N'$; then $(N', t - t', N') \in \iota$.
   (b) The inverse of a strong bounded timed bisimulation is a strong bounded timed bisimulation.
      i. Assume $(N_1, t, N_2) \in \mathcal{R}^{-1}$, namely $(N_2, t, N_1) \in \mathcal{R}$. Consider $N_2 \xrightarrow{\lambda} N_2'$; then for some N_1' we have $N_1 \xrightarrow{\lambda} N_1'$ and $(N_2', t, N_1') \in \mathcal{R}$, namely $(N_1', t, N_2') \in \mathcal{R}^{-1}$. By similar reasoning, if $N_1 \xrightarrow{\lambda} N_1'$ then we can find N_2' such that $N_2 \xrightarrow{\lambda} N_2'$ and $(N_1', t, N_2') \in \mathcal{R}^{-1}$.
      ii. Assume $(N_1, t, N_2) \in \mathcal{R}^{-1}$, namely $(N_2, N_1) \in \mathcal{R}$. Consider $N_2 \overset{t'}{\rightsquigarrow} N_2'$; then for some N_1' we have $N_1 \overset{t'}{\rightsquigarrow} N_1'$ and $(N_2', t - t', N_1') \in \mathcal{R}$, namely $(N_1', t - t', N_2') \in \mathcal{R}^{-1}$. By similar reasoning, if $N_1 \overset{t'}{\rightsquigarrow} N_1'$ then we can find N_2' such that $N_2 \overset{t'}{\rightsquigarrow} N_2'$ and $(N_1', t - t', N_2') \in \mathcal{R}^{-1}$.
   (c) The composition of strong bounded timed bisimulations is a strong bounded timed bisimulation.
      i. Assume $(N_1, t, N_2) \in \mathcal{R}_1\mathcal{R}_2$. Then for some N we have $(N_1, t, N) \in \mathcal{R}_1$ and $(N, t, N_2) \in \mathcal{R}_2$. Consider $N_1 \xrightarrow{\lambda} N_1'$; then for some N', since $(N_1, t, N) \in \mathcal{R}_1$, we have $N \xrightarrow{\lambda} N'$ and $(N_1', t, N') \in \mathcal{R}_1$. Also, since $(N, t, N_2) \in \mathcal{R}_2$ we have for some N_2' that $N_2 \xrightarrow{\lambda} N_2'$ and $(N', t, N_2') \in \mathcal{R}_2$. Thus, $(N_1', t, N_2') \in \mathcal{R}_1\mathcal{R}_2$. By similar reasoning, if $N_2 \xrightarrow{\lambda} N_2'$ then we can find N_1' such that $N_1 \xrightarrow{\lambda} N_1'$ and $(N', t, N_2') \in \mathcal{R}_2$.

ii. Assume $(N_1, t, N_2) \in \mathcal{R}_1\mathcal{R}_2$. Then for some N we have $(N_1, t, N) \in \mathcal{R}_1$ and $(N, t, N_2) \in \mathcal{R}_2$. Consider $N_1 \overset{t'}{\rightsquigarrow} N_1'$; then for some N', since $(N_1, t, N) \in \mathcal{R}_1$, we have $N \overset{t'}{\rightsquigarrow} N'$ and $(N_1', t - t', N') \in \mathcal{R}_1$. Also, since $(N, t, N_2) \in \mathcal{R}_2$, for some N_2' we have $N_2 \overset{t'}{\rightsquigarrow} N_2'$ and $(N', t - t', N_2') \in \mathcal{R}_2$. Thus, $(N_1' t - t', N_2') \in \mathcal{R}_1\mathcal{R}_2$. By similar reasoning, if $N_2 \overset{t'}{\rightsquigarrow} N_2'$ then we can find N_1' such that $N_1 \overset{t'}{\rightsquigarrow} N_1'$ and $(N', t - t', N_2') \in \mathcal{R}_2$.

(d) The union of strong bounded timed bisimulations is a strong bounded timed bisimulation.

i. Assume $(N_1, t, N_2) \in \bigcup_{i\in I}\mathcal{R}_i$. Then for some $i \in I$ we have that $(N_1, t, N_2) \in \mathcal{R}_i$. Consider $N_1 \xrightarrow{\lambda} N_1'$; then for some N_2', since $(N_1, t, N_2) \in \mathcal{R}_i$, we have $N_2 \xrightarrow{\lambda} N_2'$ and $(N_1', t, N_2') \in \mathcal{R}_i$. Thus, $(N_1', t, N_2') \in \bigcup_{i\in I}\mathcal{R}_i$. By similar reasoning, if $N_2 \xrightarrow{\lambda} N_2'$ then we can find N_1' such that $N_1 \xrightarrow{\lambda} N_1'$ and $(N_1', t, N_2') \in \mathcal{R}_i$, namely $(N_1', t, N_2') \in \bigcup_{i\in I}\mathcal{R}_i$.

ii. Assume $(N_1, t, N_2) \in \bigcup_{i\in I}\mathcal{R}_i$. Then for some $i \in I$ we have that $(N_1, t, N_2) \in \mathcal{R}_i$. Consider $N_1 \overset{t'}{\rightsquigarrow} N_1'$; then for some N_2', since $(N_1, t, N_2) \in \mathcal{R}_i$, we have $N_2 \overset{t'}{\rightsquigarrow} N_2'$ and $(N_1', t - t', N_2') \in \mathcal{R}_i$. Thus, $(N_1', t - t', N_2') \in \bigcup_{i\in I}\mathcal{R}_i$. By similar reasoning, if $N_2 \overset{t'}{\rightsquigarrow} N_2'$ then we can find N_1' such that $N_1 \overset{t'}{\rightsquigarrow} N_1'$ and $(N_1', t - t', N_2') \in \mathcal{R}_i$, namely $(N_1', t - t', N_2') \in \bigcup_{i\in I}\mathcal{R}_i$.

2. By the previous case (the union part), $\simeq$ is a strong bounded timed bisimulation and includes any other strong bounded timed bisimulation.
3. Proving that relation $\simeq$ is a timed equivalence requires proving that it satisfies reflexivity, symmetry and transitivity. We consider each of them in what follows:
   (a) Reflexivity: For any network N, $N \simeq N$ results from the fact that the identity relation is a strong bounded timed bisimulation.
   (b) Symmetry: If $N \simeq N'$, then $(N, t, N') \in \mathcal{R}$ for some strong bounded timed bisimulation $\mathcal{R}$. Hence $(N', t, N) \in \mathcal{R}^{-1}$, and so $N' \simeq N$ because the inverse relation is a strong bounded timed bisimulation.
   (c) Transitivity: If $N \simeq N'$ and $N' \simeq N''$ then $(N, t, N') \in \mathcal{R}_1$ and $(N', t, N'') \in \mathcal{R}_2$ for some strong bounded timed bisimulations $\mathcal{R}_1$ and $\mathcal{R}_2$. Thus, it holds that $(N, t, N'') \in \mathcal{R}_1\mathcal{R}_2$, and so $N \simeq N''$ due to the fact that the composition relation is a strong bounded timed bisimulation.
4. We provide Example 3 below that illustrates the strict inclusion. □

The next result claims that strong bounded timed equivalence $\simeq_t$ over processes is preserved even if the local knowledge of the agents is expanded. This is consistent with the fact that the processes affect the same portion of their knowledge.

Proposition 7. *If $K_{ij}' \subseteq K_{ij}''$, for $1 \leq i \leq n$, $1 \leq j \leq m$, then*
$$|_{i=1}^{n}\, l_i[[|]_{j=1}^{m}\, P_{ij} \rhd K_{ij}']] \;\simeq\; |_{i=1}^{n}\, l_i[[|]_{j=1}^{m}\, P_{ij} \rhd K_{ij}'']].$$

Proof. We show that $\mathcal{S}$ is a strong bounded timed bisimulation, where:
$\mathcal{S} = \{(|_{i=1}^{n}\, l_i[[|]_{j=1}^{m}\, P_{ij} \rhd K_{ij}']]\, , t,\ |_{i=1}^{n}\, l_i[[|]_{j=1}^{m}\, P_{ij} \rhd K_{ij}'']]) : K_{ij}' \subseteq K_{ij}'', 1 \leq i \leq n, 1 \leq j \leq m\}$.
The proof is by induction on the last performed step:

- Let us assume that $|_{i=1}^{n}\, l_i[[|]_{j=1}^{m}\, P_{ij} \rhd K_{ij}']] \xrightarrow{\lambda} N'$. Depending on the value of λ, there are several cases:

- Consider $\lambda = a!?@l_1$. Then there exists $P_{11} = a^{\Delta t_1}!\langle v\rangle$ then P'_{11} else P''_{11} and $P_{11} = a^{\Delta t_2}?(u)$ then P'_{12} else P''_{12} such that $l_1[[P_{11} \rhd K'_{11} \,||\, P_{12} \rhd K'_{12} \,||_{j=3}^{m}\, P_{1j} \rhd K'_{1j}]] \,|_{i=2}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K'_{ij}]] \xrightarrow{a!?@l_1} l_1[[P'_{11} \rhd K'_{11} \;||\; P'_{12} \rhd K'_{12} \;||_{j=3}^{m}\; P_{1j} \rhd K'_{1j}]] \,|_{i=2}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K'_{ij}]] = N'$. Then there exists $N'' = [[P'_{11} \rhd K''_{11} \,||\, P'_{12} \rhd K''_{12} \,||_{j=3}^{m}\, P_{1j} \rhd K''_{1j}]] \,|_{i=2}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K''_{ij}]]$ such that $|_{i=1}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K''_{ij}]] \xrightarrow{a!?@l_1} N''$. Since $K'_{ij} \subseteq K''_{ij}, 1 \le i \le n, 1 \le j \le m$, clearly $(N', t, N'') \in \mathcal{S}$.
- Consider $\lambda = a!^{\Delta 0}@l_1$. Then there exists $P_{11} = a^{\Delta 0}!\langle v\rangle$ then P'_{11} else P''_{11} such that $l_1[[P_{11} \rhd K'_{11} \,||_{j=2}^{m}\, P_{1j} \rhd K'_{1j}]] \,|_{i=2}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K'_{ij}]] \xrightarrow{a!^{\Delta 0}@l_1} l_1[[P'_{11} \rhd K'_{11} \,||_{j=2}^{m}\, P_{1j} \rhd K'_{1j}]] \,|_{i=2}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K'_{ij}]] = N'$. Then there exists $N'' = l_1[[P'_{11} \rhd K''_{11} \,||_{j=2}^{m}\, P_{1j} \rhd K''_{1j}]] \,|_{i=2}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K''_{ij}]]$ such that $|_{i=1}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K''_{ij}]]) \xrightarrow{a!^{\Delta 0}@l_1} N''$. Since $K'_{ij} \subseteq K''_{ij}, 1 \le i \le n, 1 \le j \le m$, clearly $(N', t, N'') \in \mathcal{S}$.
- Consider $\lambda = a?^{\Delta 0}@l_1$. Then there exists $P_{11} = a^{\Delta 0}?(u)$ then P'_{11} else P''_{11} such that $l_1[[P_{11} \rhd K'_{11} \,||_{j=2}^{m}\, P_{1j} \rhd K'_{1j}]] \,|_{i=2}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K'_{ij}]] \xrightarrow{a?^{\Delta 0}@l_1} l_1[[P'_{11} \rhd K'_{11} \,||_{j=2}^{m}\, P_{1j} \rhd K'_{1j}]] \,|_{i=2}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K'_{ij}]] = N'$. Then there exists $N'' = l_1[[P'_{11} \rhd K''_{11} \,||_{j=2}^{m}\, P_{1j} \rhd K''_{1j}]] \,|_{i=2}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K''_{ij}]]$ such that $|_{i=1}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K''_{ij}]]) \xrightarrow{a?^{\Delta 0}@l_1} N''$. Since $K'_{ij} \subseteq K''_{ij}, 1 \le i \le n, 1 \le j \le m$, clearly $(N', t, N'') \in \mathcal{S}$.
- Consider $\lambda = l_1 \triangleright l_2$. Then there exists $P_{11} = \mathsf{go}^0\, l_2$ then P'_{11} such that $l_1[[P_{11} \rhd K'_{11} \,||_{j=2}^{m}\, P_{1j} \rhd K'_{1j}]] \,|_{i=2}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K'_{ij}]] \xrightarrow{l_1 \triangleright l_2} l_1[[||_{j=2}^{m}\, P_{1j} \rhd K'_{1j}]] \mid l_2[[P'_{11} \rhd K'_{11} \,||_{j=1}^{m}\, P_{2j} \rhd K'_{2j}]] \,|_{i=3}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K'_{ij}]] = N'$. Then there exists $N'' = l_1[[||_{j=2}^{m}\, P_{1j} \rhd K''_{1j}]] \mid l_2[[P'_{11} \rhd K''_{11} \,||_{j=1}^{m}\, P_{2j} \rhd K''_{2j}]] \,|_{i=3}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K''_{ij}]]$ such that $|_{i=1}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K''_{ij}]]) \xrightarrow{l_1 \triangleright l_2} N''$. Since $K'_{ij} \subseteq K''_{ij}, 1 \le i \le n, 1 \le j \le m$, clearly $(N', t, N'') \in \mathcal{S}$.
- Consider $\lambda = \mathit{true}@l_1$. Then there exists $P_{11} = \mathsf{if}\ \mathit{test}\ \mathsf{then}\ P'_{11}\ \mathsf{else}\ P''_{11}$, where $\mathit{test}@K'_{11} = \mathit{true}$, such that $l_1[[P_{11} \rhd K'_{11} \,||_{j=2}^{m}\, P_{1j} \rhd K'_{1j}]] \,|_{i=2}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K'_{ij}]] \xrightarrow{\mathit{true}@l_1} l_1[[P'_{11} \rhd K'_{11} \,||_{j=2}^{m}\, P_{1j} \rhd K'_{1j}]] \,|_{i=2}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K'_{ij}]] = N'$. Then there exists $N'' = l_1[[P'_{11} \rhd K''_{11} \,||_{j=2}^{m}\, P_{1j} \rhd K''_{1j}]] \,|_{i=2}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K''_{ij}]]$ such that $|_{i=1}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K''_{ij}]]) \xrightarrow{\mathit{true}@l_1} N''$. Since $K'_{ij} \subseteq K''_{ij}, 1 \le i \le n, 1 \le j \le m$, clearly $(N', t, N'') \in \mathcal{S}$.
- Consider $\lambda = \mathit{false}@l_1$. Then there exists $P_{11} = \mathsf{if}\ \mathit{test}\ \mathsf{then}\ P'_{11}\ \mathsf{else}\ P''_{11}$, where $\mathit{test}@K'_{11} = \mathit{false}$, such that $l_1[[P_{11} \rhd K'_{11} \,||_{j=2}^{m}\, P_{1j} \rhd K'_{1j}]] \,|_{i=2}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K'_{ij}]] \xrightarrow{\mathit{false}@l_1} l_1[[P''_{11} \rhd K'_{11} \,||_{j=2}^{m}\, P_{1j} \rhd K'_{1j}]] \,|_{i=2}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K'_{ij}]] = N'$. Then there exists $N'' = l_1[[P''_{11} \rhd K''_{11} \,||_{j=2}^{m}\, P_{1j} \rhd K''_{1j}]] \,|_{i=2}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K''_{ij}]]$ such that $|_{i=1}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K''_{ij}]]) \xrightarrow{\mathit{false}@l_1} N''$. Since $K'_{ij} \subseteq K''_{ij}, 1 \le i \le n, 1 \le j \le m$, clearly $(N', t, N'') \in \mathcal{S}$.
- Consider $\lambda = \mathit{create}_f@l_1$. Then there exists $P_{11} = \mathit{create}(\langle f \mid v; \emptyset\rangle)$ then P'_{11} such that $l_1[[P_{11} \rhd K'_{11} \,||_{j=2}^{m}\, P_{1j} \rhd K'_{1j}]] \,|_{i=2}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K'_{ij}]] \xrightarrow{\mathit{create}_f@l_1} l_1[[P'_{11} \rhd K'_{11} \;\langle f \mid v; \emptyset\rangle \,||_{j=2}^{m}\, P_{1j} \rhd K'_{1j}]] \,|_{i=2}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K'_{ij}]] = N'$. Then there exists $N'' = l_1[[P'_{11} \rhd K''_{11} \;\langle f \mid v; \emptyset\rangle \,||_{j=2}^{m}\, P_{1j} \rhd K''_{1j}]] \,|_{i=2}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K''_{ij}]]$ such that $|_{i=1}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K''_{ij}]]) \xrightarrow{\mathit{create}_f@l_1} N''$. Since $K'_{ij} \subseteq K''_{ij}, 1 \le i \le n, 1 \le j \le m$, then also $K'_{11}\ \langle f \mid v; \emptyset\rangle \subseteq K''_{11}\ \langle f \mid v; \emptyset\rangle$, and clearly $(N', t, N'') \in \mathcal{S}$.
- Consider $\lambda = \mathit{upd}_p@l_1$. Then there exists $P_{11} = \mathit{update}(p/f, v)$ then P'_{11} such that $l_1[[P_{11} \rhd K'_{11} \,||_{j=2}^{m}\, P_{1j} \rhd K'_{1j}]] \,|_{i=2}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K'_{ij}]] \xrightarrow{\mathit{upd}_p@l_1} l_1[[P'_{11} \rhd K^{u'}_{11} \,||_{j=2}^{m}\, P_{1j} \rhd K'_{1j}]] \,|_{i=2}^{n}\, l_i[[||_{j=1}^{m}\, P_{ij} \rhd K'_{ij}]] = N'$. Then there exists $N'' = l_1[[P'_{11} \rhd K^{u''}_{11}$

$||_{j=2}^{m}\ P_{1j} \rhd K'_{1j}]]\ \ |_{i=2}^{n}\ l_i[[||_{j=1}^{m}\ P_{ij} \rhd K'_{ij}]]]$ such that $|_{i=1}^{n}\ l_i[[||_{j=1}^{m}\ P_{ij} \rhd K''_{ij}]] \xrightarrow{upd_p@l_1} N''$. Since $K'_{ij} \subseteq K''_{ij}$, $1 \leq i \leq n, 1 \leq j \leq m$, then also $K_{11}^{u'} \subseteq K_{11}^{u''}$, and clearly $(N', t, N'') \in \mathcal{S}$.

- Let us assume that $|_{i=1}^{n}\ l_i[[||_{j=1}^{m}\ P_{ij} \rhd K'_{ij}]] \overset{t'}{\rightsquigarrow} N'$. Then there exists P'_{ij}, $1 \leq i \leq n$, $1 \leq j \leq m$, such that $|_{i=1}^{n}\ l_i[[||_{j=1}^{m}\ P_{ij} \rhd K'_{ij}]] \overset{t'}{\rightsquigarrow} |_{i=1}^{n}\ l_i[[||_{j=1}^{m}\ P'_{ij} \rhd K'_{ij}]] = N'$. Then there exists $N'' = |_{i=1}^{n}\ l_i[[||_{j=1}^{m}\ P'_{ij} \rhd K''_{ij}]]$ such that $|_{i=1}^{n}\ l_i[[||_{j=1}^{m}\ P_{ij} \rhd K''_{ij}]] \overset{t}{\rightsquigarrow} N''$. Since $K'_{ij} \subseteq K''_{ij}$, $1 \leq i \leq n$, $1 \leq j \leq m$, clearly $(N', t - t', N'') \in \mathcal{S}$.
  The symmetric cases follow by similar arguments. □

The following result shows that strong bounded timed bisimulation is preserved even after complete computational steps of two networks in knowTiMo.

Proposition 8. *Let N_1, N_2 be two* knowTiMo *networks.*
If $N_1 \simeq_t N_2$ and $N_1 \overset{\Lambda,t'}{\Longrightarrow} N'_1$, then there is $N'_2 \in \mathcal{N}$ such that $N_2 \overset{\Lambda,t'}{\Longrightarrow} N'_2$ and $N'_1 \simeq_{t-t'} N'_2$.

Proof. Assuming that the finite multiset of actions Λ contains the labels $\{\lambda_1, \ldots, \lambda_k\}$, then the complete computational step $N_1 \overset{\Lambda,t}{\Longrightarrow} N'_1$ can be detailed as $N_1 \xrightarrow{\lambda_1} N_1^1 \ldots N_1^{k-1} \xrightarrow{\lambda_k} N_1^k \overset{t'}{\rightsquigarrow} N'_1$. Note that $N_1 \simeq_t N_2$ means that $(N_1, t, N_2) \in \simeq$. Since $N_1 \xrightarrow{\lambda_1} N_1^1$ and $(N_1, t, N_2) \in \simeq$, then according to Definition 2 there exists $N_2^1 \in \mathcal{N}$ such that $N_2 \xrightarrow{\lambda_1} N_2^1$ and $(N_1^1, t, N_2^1) \in \simeq$. The same reasoning can be applied for another k steps, meaning that there exist $N_2^2, \ldots, N_2^k, N'_2 \in \mathcal{N}$ such that $N_2 \xrightarrow{\lambda_1} N_2^1 \ldots N_2^{k-1} \xrightarrow{\lambda_k} N_2^k \overset{t'}{\rightsquigarrow} N'_2$ and $(N'_1, t - t', N'_2) \in \simeq$, namely $N'_1 \simeq_{t-t'} N'_2$. The definition of a complete computational step implies that $N_2 \xrightarrow{\lambda_1} N_2^1 \ldots N_2^{k-1} \xrightarrow{\lambda_k} N_2^k \overset{t'}{\rightsquigarrow} N'_2$ can be written as $N_2 \overset{\Lambda,t'}{\Longrightarrow} N'_2$. Thus, we obtained that there exists $N'_2 \in \mathcal{N}$ such that $N_2 \overset{\Lambda,t'}{\Longrightarrow} N'_2$ and $N'_1 \simeq_{t-t'} N'_2$ (as desired). □

Strong bounded timed bisimulation satisfies the property that if two networks are equivalent up-to a certain deadline t, they are equivalent up-to any deadline t' before t, i.e., $t' \leq t$.

Proposition 9. *If $N \simeq_t N'$ and $t' \leq t$, then $N \simeq_{t'} N'$.*

Proof. Assume $N \simeq_t N'$ and that there exist the networks $N_1, \ldots, N_k \in \mathcal{N}$, the set of actions $\Lambda_1, \ldots, \Lambda_k$ and the timers $t_1, \ldots, t_k \in \mathbb{N}$ such that $N \overset{\Lambda_1,t_1}{\Longrightarrow} N_1 \ldots \overset{\Lambda_k,t_k}{\Longrightarrow} N_k$ and also $t = t_1 + \ldots + t_k$. According to Proposition 8, there exist the networks $N'_1, \ldots, N'_k \in \mathcal{N}$ such that $N' \overset{\Lambda_1,t_1}{\Longrightarrow} N'_1 \ldots \overset{\Lambda_k,t_k}{\Longrightarrow} N'_k$, and also $N_1 \simeq_{t-t_1} N'_1, \ldots, N_k \simeq_0 N'_k$. Since $t' \leq t$, then there exists an $l \leq k$ and a $t'' \in \mathbb{N}$ such that $t_1 + \ldots + t_l + t'' = t'$. By using Theorem 1, it holds that there exists N^1 such that $N \overset{\Lambda_1,t_1}{\Longrightarrow} N_1 \ldots \overset{\Lambda_l,t_l}{\Longrightarrow} N_l \overset{\Lambda_{l+1},t''}{\Longrightarrow} N^1$. In a similar manner, by using Theorem 1, it holds that there exists N^2 such that $N' \overset{\Lambda_1,t_1}{\Longrightarrow} N'_1 \ldots \overset{\Lambda_l,t_l}{\Longrightarrow} N'_l \overset{\Lambda_{l+1},t''}{\Longrightarrow} N^2$. Since N^1 and N_2 can perform only time passing steps of length at most $t_{l+1} - t''$, this means that $N^1 \simeq_0 N^2$, However, according to Definition 2, this means that we obtain the desired relation $N \simeq_{t'} N'$ because the networks N and N' can match their behaviour for t' steps. □

The next example illustrates that the relation $\simeq_t$ is able to treat as bisimilar some multi-agent systems that are not bisimilar using the relation $\sim$.

Example 3. *Let us consider the networks of Example 2, namely:*

$N_1 = \text{office}_4[[\text{C}'_2(\text{office}_4) \triangleright K_{\text{C2}}]]$,
$N'_1 = \text{office}_4[[\text{C}'_2(\text{office}_4) \triangleright K'_{\text{C2}}]]$,
$N''_1 = \text{office}_4[[\text{C}'_2(\text{office}_4) \triangleright K''_{\text{C2}}]]$,

where the knowledge of the agents is defined as:

$K_{\text{C2}} = \langle \mathit{agency} \mid \text{office}_4; \langle \mathit{dest} \mid \varepsilon \rangle \langle \mathit{price} \mid \varepsilon \rangle \rangle$,
$K'_{\text{C2}} = \langle \mathit{agency} \mid \text{office}_5; \langle \mathit{dest} \mid \varepsilon \rangle \langle \mathit{price} \mid \varepsilon \rangle \rangle$,
$K''_{\text{C2}} = \emptyset$.

Even if it holds that $N'_1 \sim N''_1$ while $N_1 \not\sim N'_1$ and $N_1 \not\sim N''_1$, by applying Definition 2, it results that N_1, N'_1 and N''_1 are strong bounded timed bisimilar before the 4th time unit since they have the same evolutions during this deadline, namely $N'_1 \simeq_4 N''_1$, $N_1 \simeq_4 N'_1$ and $N_1 \simeq_4 N''_1$. If $t > 4$, we have that $N'_1 \simeq_t N''_1$, while $N_1 \not\simeq_t N'_1$ and $N_1 \not\simeq_t N''_1$. Thus, both Definitions 1 and 2 return the same relations among N_1, N'_1 and N''_1 for $t > 4$.

This example illustrates also the strict inclusion relation from item 4 of Proposition 6.

3.3. Weak Knowledge Equivalences

Both equivalence relations $\sim$ and $\simeq$ require an exact match of transitions and time steps of two networks in knowTiMo; this makes them too restrictive. We can introduce a weaker version of network equivalence by looking only at the steps that affect the knowledge data, namely the *create* and *update* steps. Thus, we introduce a knowledge equivalence in order to distinguish between networks based on the interaction of the agents with their local knowledge: the networks are equivalent if we observe only *create* and *update* actions along same paths, regardless of the values added to the knowledge.

Definition 3 (Weak knowledge bisimulation)**.** *Let $\mathcal{R} \subseteq \mathcal{N} \times \mathcal{N}$ be a symmetric binary relation over networks in* knowTiMo*.*

1. *$\mathcal{R}$ is a* weak knowledge bisimulation *if*
   - *$(N_1, N_2) \in \mathcal{R}$ and $N_1 \xrightarrow{create_f@l}{}^* N'_1$ implies that there exists $N'_2 \in \mathcal{N}$ such that $N_2 \xrightarrow{create_f@l}{}^* N'_2$ and $(N'_1, N'_2) \in \mathcal{R}$;*
   - *$(N_1, N_2) \in \mathcal{R}$ and $N_1 \xrightarrow{upd_p@l}{}^* N'_1$ implies that there exists $N'_2 \in \mathcal{N}$ such that $N_2 \xrightarrow{upd_p@l}{}^* N'_2$ and $(N'_1, N'_2) \in \mathcal{R}$;*
2. *The* weak knowledge bisimilarity *is the union $\cong$ of all weak knowledge bisimulations $\mathcal{R}$.*

The following results present some properties of the weak knowledge bisimulations. In particular, we prove that the equivalence relation $\cong$ (that is strictly included in relation $\sim$) is the largest weak knowledge bisimulation.

Proposition 10.

1. *Identity, inverse, composition and union of weak knowledge bisimulations are weak knowledge bisimulations.*
2. *$\cong$ is the largest weak knowledge bisimulation.*
3. *$\cong$ is an equivalence.*
4. *$\cong \subsetneq \sim$.*

Proof.

1. We treat each relation separately showing that it respects the conditions from Definition 3 for being a weak knowledge bisimulation.
   (a) The identity relation $Id_{\mathcal{R}}$ is a weak knowledge bisimulation.
      i. Assume $(N, N) \in Id_{\mathcal{R}}$. Consider $N \xrightarrow{create_f@l}{}^* N'$; then $(N', N') \in Id_{\mathcal{R}}$.

ii. Assume $(N, N) \in Id_{\mathcal{R}}$. Consider $N \xRightarrow{upd_p@l} N'$; then $(N', N') \in Id_{\mathcal{R}}$.

(b) The inverse of a weak knowledge bisimulation is a weak knowledge bisimulation.

i. Assume $(N_1, N_2) \in \mathcal{R}^{-1}$, namely $(N_2, N_1) \in \mathcal{R}$. Consider $N_2 \xRightarrow{create_f@l} N_2'$; then for some N_1' we have $N_1 \xRightarrow{create_f@l} N_1'$ and $(N_2', N_1') \in \mathcal{R}$, namely $(N_1', N_2') \in \mathcal{R}^{-1}$. By similar reasoning, if $N_1 \xRightarrow{create_f@l} N_1'$ then we can find N_2' such that $N_2 \xRightarrow{create_f@l} N_2'$ and $(N_1', N_2') \in \mathcal{R}^{-1}$.

ii. Assume $(N_1, N_2) \in \mathcal{R}^{-1}$, namely $(N_2, N_1) \in \mathcal{R}$. Consider $N_2 \xRightarrow{upd_f@l} N_2'$; then for some N_1' we have $N_1 \xRightarrow{upd_f@l} N_1'$ and $(N_2', N_1') \in \mathcal{R}$, namely $(N_1', N_2') \in \mathcal{R}^{-1}$. By similar reasoning, if $N_1 \xRightarrow{upd_f@l} N_1'$ then we can find N_2' such that $N_2 \xRightarrow{upd_f@l} N_2'$ and $(N_1', N_2') \in \mathcal{R}^{-1}$.

(c) The composition of weak knowledge bisimulations is a weak knowledge bisimulation.

i. Assume $(N_1, N_2) \in \mathcal{R}_1\mathcal{R}_2$. Then for some N we have $(N_1, N) \in \mathcal{R}_1$ and $(N, N_2) \in \mathcal{R}_2$. Consider $N_1 \xRightarrow{create_f@l} N_1'$; then for some N', since $(N_1, N) \in \mathcal{R}_1$, we have $N \xRightarrow{create_f@l} N'$ and $(N_1', N') \in \mathcal{R}_1$. Also, since $(N, N_2) \in \mathcal{R}_2$ we have for some N_2' that $N_2 \xRightarrow{create_f@l} N_2'$ and $(N', N_2') \in \mathcal{R}_2$. Thus, $(N_1', N_2') \in \mathcal{R}_1\mathcal{R}_2$. By similar reasoning, if $N_2 \xRightarrow{create_f@l} N_2'$ then we can find N_1' such that $N_1 \xRightarrow{create_f@l} N_1'$ and $(N', N_2') \in \mathcal{R}_2$.

ii. Assume $(N_1, N_2) \in \mathcal{R}_1\mathcal{R}_2$. Then for some N we have $(N_1, N) \in \mathcal{R}_1$ and $(N, N_2) \in \mathcal{R}_2$. Consider $N_1 \xRightarrow{upd_f@l} N_1'$; then for some N', since $(N_1, N) \in \mathcal{R}_1$, we have $N \xRightarrow{upd_f@l} N'$ and $(N_1', N') \in \mathcal{R}_1$. Also, since $(N, N_2 \in \mathcal{R}_2$ we have for some N_2' that $N_2 \xRightarrow{upd_f@l} N_2'$ and $(N', N_2') \in \mathcal{R}_2$. Thus, $(N_1', N_2') \in \mathcal{R}_1\mathcal{R}_2$. By similar reasoning, if $N_2 \xRightarrow{upd_f@l} N_2'$ then we can find N_1' such that $N_1 \xRightarrow{upd_f@l} N_1'$ and $(N', N_2') \in \mathcal{R}_2$.

(d) The union of weak knowledge bisimulations is a weak knowledge bisimulation.

i. Assume $(N_1, N_2) \in \bigcup_{i \in I} \mathcal{R}_i$. Then for some $i \in I$ we have $(N_1, N_2) \in \mathcal{R}_i$. Consider $N_1 \xRightarrow{create_f@l} N_1'$; then for some N_2', since $(N_1, N_2) \in \mathcal{R}_i$, we have $N_2 \xRightarrow{create_f@l} N_2'$ and $(N_1', N_2') \in \mathcal{R}_i$. Thus, $(N_1', N_2') \in \bigcup_{i \in I} \mathcal{R}_i$. By similar reasoning, if $N_2 \xRightarrow{create_f@l} N_2'$ then we can find N_1' such that $N_1 \xRightarrow{create_f@l} N_1'$ and $(N_1', N_2') \in \mathcal{R}_i$, namely $(N_1', N_2') \in \bigcup_{i \in I} \mathcal{R}_i$.

ii. Assume $(N_1, N_2) \in \bigcup_{i \in I} \mathcal{R}_i$. Then for some $i \in I$ we have $(N_1, N_2) \in \mathcal{R}_i$. Consider $N_1 \xRightarrow{upd_f@l} N_1'$; then for some N_2', since $(N_1, N_2) \in \mathcal{R}_i$, we have $N_2 \xRightarrow{upd_f@l} N_2'$ and $(N_1', N_2') \in \mathcal{R}_i$. Thus, $(N_1', N_2') \in \bigcup_{i \in I} \mathcal{R}_i$. By similar reasoning, if $N_2 \xRightarrow{upd_f@l} N_2'$ then we can find N_1' such that $N_1 \xRightarrow{upd_f@l} N_1'$ and $(N_1', N_2') \in \mathcal{R}_i$, namely $(N_1', N_2') \in \bigcup_{i \in I} \mathcal{R}_i$.

2. By the previous case (the union part), $\cong$ is a weak knowledge bisimulation and includes any other weak knowledge bisimulation.
3. Proving that relation $\cong$ is an equivalence requires proving that it satisfies reflexivity, symmetry and transitivity. We consider each of them in what follows:

(a) Reflexivity: For any network N, $N \cong N$ results from the fact that the identity relation is a weak knowledge bisimulation.

(b) Symmetry: If $N \cong N'$, then $(N, N') \in \mathcal{R}$ for some weak knowledge bisimulation $\mathcal{R}$. Hence $(N', N) \in \mathcal{R}^{-1}$, and so $N' \cong N$ because the inverse relation is a weak knowledge bisimulation.

(c) Transitivity: If $N \cong N'$ and $N' \cong N''$ then $(N, N') \in \mathcal{R}_1$ and $(N', N'') \in \mathcal{R}_2$ for some weak knowledge bisimulations $\mathcal{R}_1$ and $\mathcal{R}_2$. Thus, $(N, N'') \in \mathcal{R}_1\mathcal{R}_2$, and so $N \cong N''$ due to the fact that the composition relation is a weak knowledge bisimulation.

4. We provide Example 4 below illustrating the strict inclusion. □

The next result claims that weak knowledge equivalence $\cong$ among processes is preserved even if the local knowledge of the agents is expanded. This is consistent with the fact that the processes affect the same portion of their knowledge.

Proposition 11. *If $K'_{ij} \subseteq K''_{ij}$, for $1 \leq i \leq n$, $1 \leq j \leq m$, then*
$$|_{i=1}^{n}\, l_i[[|]_{j=1}^{m}\, P_{ij} \rhd K'_{ij}]] \;\cong\; |_{i=1}^{n}\, l_i[[|]_{j=1}^{m}\, P_{ij} \rhd K''_{ij}]].$$

Proof. We show that $\mathcal{S}$ is a weak knowledge bisimulation, where:
$\mathcal{S} = \{(|_{i=1}^{n}\, l_i[[|]_{j=1}^{m}\, P_{ij} \rhd K'_{ij}]\ ,\ |_{i=1}^{n}\, l_i[[|]_{j=1}^{m}\, P_{ij} \rhd K''_{ij}]]) \ :\ K'_{ij} \subseteq K''_{ij}, 1 \leq i \leq n, 1 \leq j \leq m\}$.
The proof is by induction on the last performed step. Let us assume that $|_{i=1}^{n}\, l_i[[|]_{j=1}^{m}\, P_{ij} \rhd K'_{ij}]] \stackrel{\lambda}{\Longrightarrow}{}^{*} N'$. Depending on the value of λ, there are several cases:

- Consider $\lambda = create_f@l_1$. Then there exists P_{11} such that $l_1[[P_{11} \rhd K'_{11}\ ||_{j=2}^{m}\ P_{1j} \rhd K'_{1j}]]\ |_{i=2}^{n}\ l_i[[|]_{j=1}^{m}\ P_{ij} \rhd K'_{ij}]] \stackrel{create_f@l}{\Longrightarrow}{}^{*} l_1[[P'_{11} \rhd K'_{11}\ \langle f \mid v; \emptyset\rangle\ ||_{j=2}^{m}\ P'_{1j} \rhd K'_{1j}]]\ |_{i=2}^{n}\ l_i[[|]_{j=1}^{m}\ P'_{ij} \rhd K'_{ij}]] = N'$. Then there exists $N'' = l_1[[P'_{11} \rhd K''_{11}\ \langle f \mid v; \emptyset\rangle\ ||_{j=2}^{m}\ P'_{1j} \rhd K''_{1j}]]\ |_{i=2}^{n}\ l_i[[|]_{j=1}^{m}\ P'_{ij} \rhd K''_{ij}]]$ such that $|_{i=1}^{n}\ l_i[[|]_{j=1}^{m}\ P_{ij} \rhd K''_{ij}]]) \stackrel{create_f@l}{\Longrightarrow}{}^{*} N''$. Since $K'_{ij} \subseteq K''_{ij}$, $1 \leq i \leq n, 1 \leq j \leq m$, then also $K'_{11}\ \langle f \mid v; \emptyset\rangle \subseteq K''_{11}\ \langle f \mid v; \emptyset\rangle$, and clearly $(N', N'') \in \mathcal{S}$.
- Consider $\lambda = upd_p@l_1$. Then there exists P_{11} such that $l_1[[P_{11} \rhd K'_{11}\ ||_{j=2}^{m}\ P_{1j} \rhd K'_{1j}]]\ |_{i=2}^{n}\ l_i[[|]_{j=1}^{m}\ P_{ij} \rhd K'_{ij}]] \stackrel{upd_f@l}{\Longrightarrow}{}^{*} l_1[[P'_{11} \rhd K^{u'}_{11}\ ||_{j=2}^{m}\ P'_{1j} \rhd K'_{1j}]]\ |_{i=2}^{n}\ l_i[[|]_{j=1}^{m}\ P'_{ij} \rhd K'_{ij}]] = N'$. Then there exists $N'' = l_1[[P'_{11} \rhd K^{u''}_{11}\ ||_{j=2}^{m}\ P'_{1j} \rhd K'_{1j}]]\ |_{i=2}^{n}\ l_i[[|]_{j=1}^{m}\ P'_{ij} \rhd K'_{ij}]]]$ such that $|_{i=1}^{n}\ l_i[[|]_{j=1}^{m}\ P_{ij} \rhd K''_{ij}]]) \stackrel{upd_f@l}{\Longrightarrow}{}^{*} N''$. Since $K'_{ij} \subseteq K''_{ij}$, $1 \leq i \leq n, 1 \leq j \leq m$, then also $K^{u'}_{11} \subseteq K^{u''}_{11}$, and clearly $(N', N'') \in \mathcal{S}$.

  The symmetric cases follow by similar arguments. □

The following result shows that weak knowledge bisimulation is preserved after complete computational steps of two networks in knowTiMo only if the knowledge is modified at least once during such a step.

Proposition 12. *Let N_1, N_2 be two* knowTiMo *networks and $\exists create_f@l \in \Lambda$ or $\exists upd_f@l \in \Lambda$. If $N_1 \cong N_2$ and $N_1 \stackrel{\Lambda,t}{\Longrightarrow} N'_1$, then there exists $N'_2 \in \mathcal{N}$ such that $N_2 \stackrel{\Lambda,t}{\Longrightarrow}{}'_2$ and $N'_1 \cong N'_2$.*

Proof. Assuming that the finite multiset of actions Λ contains the labels $\{\lambda_1, \ldots, \lambda_k\}$ that denote modifications to the knowledge, then the complete computational step $N_1 \stackrel{\Lambda,t}{\Longrightarrow} N'_1$ can be detailed as $N_1 \stackrel{\lambda_1}{\Longrightarrow}{}^{*} N_1^1 \ldots N_1^{k-1} \stackrel{\lambda_k}{\Longrightarrow}{}^{*} N'_1$. Since $N_1 \stackrel{\lambda_1}{\Longrightarrow}{}^{*} N_1^1$ and $N_1 \cong N_2$, then according to Definition 3 there exists $N_2^1 \in \mathcal{N}$ such that $N_2 \stackrel{\lambda_1}{\Longrightarrow}{}^{*} N_2^1$ and $N_1^1 \cong N_2^1$. The same reasoning can be applied for another k times, meaning that there exist $N_2^2, \ldots, N'_2 \in \mathcal{N}$ such that $N_2 \stackrel{\lambda_1}{\Longrightarrow}{}^{*} N_2^1 \ldots N_2^{k-1} \stackrel{\lambda_k}{\Longrightarrow}{}^{*} N'_2$ and $N'_1 \cong N'_2$. By the definition of a complete computational step, it holds that $N_2 \stackrel{\lambda_1}{\Longrightarrow}{}^{*} N_2^1 \ldots N_2^{k-1} \stackrel{\lambda_k}{\Longrightarrow}{}^{*} N'_2$ can be written as $N_2 \stackrel{\Lambda,t}{\Longrightarrow} N'_2$. Thus, we obtained that there exists $N'_2 \in \mathcal{N}$ such that $N_2 \stackrel{\Lambda,t}{\Longrightarrow} N'_2$ and $N'_1 \cong N'_2$ (as desired). □

The next example illustrates that the relation $\cong$ is able to treat bisimilar systems that are not bisimilar using the relation $\sim$.

Example 4. *Consider the network N_1 of Example 2, and a network*

$$N_1''' = \text{office}_4[[C_2''(\text{office}_4) \rhd K_{C2}]],$$

in which the client can perform only an update action:

$$C_2''(\text{office}_4) = update(/agency[test_4]/dest, dest_{C2,1}).$$

According to Definition 1, it holds that $N_1 \not\sim N_1'''$. This is due to the fact that the network N_1 can perform a time step of length 4 and choose the else *branch, while the network N_1''' can perform only the update operation. Formally:*

$$N_1 \overset{4}{\rightsquigarrow} N_2 \xrightarrow{false@\text{office}_4} N_3 \xrightarrow{upd_{/agency[test_4]/dest}@\text{office}_4} N_4$$

and

$$N_1''' \xrightarrow{upd_{/agency[test_4]/dest}@\text{office}_4} N_2'''$$

The above reductions can also be written as

$$N_1 \Longrightarrow^* N_3 \xrightarrow{upd_{/agency[test_4]/dest}@\text{office}_4} N_4,$$

and

$$N_1''' \Longrightarrow^* N_1''' \xrightarrow{upd_{/agency[test_4]/dest}@\text{office}_4} N_2'''.$$

By applying Definition 3, it results that N_1 and N_1''' are weak knowledge bisimilar because they are able to perform an update on the same path at the same location, i.e., $N_1 \cong N_1'''$.

This example is also an illustration of the strict inclusion relation from item 4 of Proposition 10.

4. Conclusions and Related Work

In multi-agent systems, knowledge is usually treated by using epistemic logics [6]; in particular, the multi-agent epistemic logic [7,8]. These epistemic logics are modal logics describing different types of knowledge, being different not only syntactically, but also in expressiveness and complexity. Essentially, they are based on two concepts: Kripke structures (to model their semantics) and logic formulas (to represent the knowledge of the agents).

The initial version of TiMo presented in [1] leads to some extensions: with access permissions in perTiMo [9], with real-time in rTiMo [10], combining TiMo and the bigraphs [11] to obtain the BigTiMo calculus [12]. However, in all these approaches an implicit knowledge is used inside the processes. In this article we defined knowTiMo to describe multi-agent systems operating according to their accumulated knowledge. Essentially, the agents get an explicit representation of the knowledge about the other agents of a distributed network in order to decide their next interactions. The knowledge is defined as sets of trees whose nodes contain pairs of labels and values; this tree representation is similar to the data representation in Petri nets with structured data [13] and Xdπ process calculus [14]. The network dynamics involving exchanges of knowledge between agents is presented by the operational semantics of this process calculus; its labelled transition system is able to capture the concurrent execution by using a multiset of actions. We proved that time passing in such a multi-agent system does not introduce any nondeterminism in the evolution of a network, and that the progression of the network is smooth (there are no time gaps). Several results are devoted to the relationship between the evolution of the agents and their knowledge.

According to [15], the notion of bisimulation was independently discovered in computer science [16,17], modal logic [18] and set theory [19,20]. Bisimulation is currently used in several domains: to test the behavioural equality of processes in concurrency [21]; to solve the state-explosion problem in model checking [22]; to index and compress semi-structured data in databases [23,24]; to solve Markov decision processes efficiently in stochastic planning [25]; to understand for some languages their expressiveness in description logics [26]; and to study the observational indistinguishability and computational

complexity on data graphs in XPath (a language extending modal logic with equality tests for data) [27]. It is worth noting that the notion of bisimulation is related to the modal equivalence in various logics of knowledge and structures presented in [28]. In some of these logics it is proved that certain forms of bisimulation correspond to modal equivalence of knowledge, and this is used to compare the logics expressivity [29,30].

Inspired by the bisimulation notion defined in computer science, in this paper we defined and studied some specific behavioural equivalences involving the network knowledge and timing constraints on communication and migration; the defined behavioural equivalences are preserved during complete computational steps of two multi-agent systems. Strong timed bisimulation takes also into account timed transitions, being able to distinguish between different systems regardless of the evolution time; strong bounded timed bisimulation imposes limits for the evolution time, including the equivalences up to any bound below that deadline. A knowledge equivalence is able to distinguish between systems based on the interaction of the agents with their local knowledge. In the literature, a related but weaker/simpler approach of knowledge bisimulation appeared in [14], where the authors used only barbs (not equivalences), looking only at the *update* steps.

Author Contributions: All authors have read and agreed to the published version of the manuscript. All authors contributed equally to this work.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare that there is no conflict of interest.

References

1. Ciobanu, G.; Koutny, M. Modelling and Verification of Timed Interaction and Migration. In Proceedings of the Fundamental Approaches to Software Engineering, 11th International Conference, FASE 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, 29 March–6 April 2008; Lecture Notes in Computer Science. Fiadeiro, J.L., Inverardi, P., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; Volume 4961, pp. 215–229. [CrossRef]
2. Abiteboul, S.; Buneman, P.; Suciu, D. *Data on the Web: From Relations to Semistructured Data and XML*; Morgan Kaufmann: Burlington, MA, USA, 1999.
3. Aman, B.; Ciobanu, G. Verification of distributed systems involving bounded-time migration. *Int. J. Crit.-Comput.-Based Syst.* **2017**, *7*, 279–301. [CrossRef]
4. Ciobanu, G. Behaviour Equivalences in Timed Distributed pi-Calculus. In *Software-Intensive Systems and New Computing Paradigms—Challenges and Visions*; Lecture Notes in Computer Science; Wirsing, M., Banâtre, J., Hölzl, M.M., Rauschmayer, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5380, pp. 190–208. [CrossRef]
5. Posse, E.; Dingel, J. Theory and Implementation of a Real-Time Extension to the *pi*-Calculus. In Proceedings of the Formal Techniques for Distributed Systems, Joint 12th IFIP WG 6.1 International Conference, FMOODS 2010 and 30th IFIP WG 6.1 International Conference, FORTE 2010, Amsterdam, The Netherlands, 7–9 June 2010; Lecture Notes in Computer Science. Hatcliff, J., Zucca, E., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6117, pp. 125–139. [CrossRef]
6. Hintikka, J. *Knowledge and Belief. An Introduction to the Logic of the Two Notions*; Cornell University Press: Ithaca, NY, USA, 1962.
7. Fagin, R.; Halpern, J.Y. Belief, Awareness, and Limited Reasoning. *Artif. Intell.* **1987**, *34*, 39–76. [CrossRef]
8. Modica, S.; Rustichini, A. Awareness and partitional information structures. *Theory Decis.* **1994**, *37*, 107–124. [CrossRef]
9. Ciobanu, G.; Koutny, M. Timed Migration and Interaction with Access Permissions. In Proceedings of the FM 2011: Formal Methods—17th International Symposium on Formal Methods, Limerick, Ireland, 20–24 June 2011; Lecture Notes in Computer Science. Butler, M.J., Schulte, W., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6664, pp. 293–307. [CrossRef]
10. Aman, B.; Ciobanu, G. Real-Time Migration Properties of rTiMo Verified in Uppaal. In Proceedings of the Software Engineering and Formal Methods—11th International Conference, SEFM 2013, Madrid, Spain, 25–27 September 2013; Lecture Notes in Computer Science. Hierons, R.M., Merayo, M.G., Bravetti, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; Volume 8137, pp. 31–45. [CrossRef]
11. Milner, R. *The Space and Motion of Communicating Agents*; Cambridge University Press: Cambridge, UK, 2009.
12. Xie, W.; Zhu, H.; Zhang, M.; Lu, G.; Fang, Y. Formalization and Verification of Mobile Systems Calculus Using the Rewriting Engine Maude. In Proceedings of the 2018 IEEE 42nd Annual Computer Software and Applications Conference, COMPSAC 2018, Tokyo, Japan, 23–27 July 2018; Reisman, S., Ahamed, S.I., Demartini, C., Conte, T.M., Liu, L., Claycomb, W.R., Nakamura, M., Tovar, E., Cimato, S., Lung, C., et al., Eds.; IEEE Computer Society: New York, NY, USA, 2018; Volume 1, pp. 213–218. [CrossRef]
13. Badouel, É.; Hélouët, L.; Morvan, C. Petri Nets with Structured Data. *Fundam. Inform.* **2016**, *146*, 35–82. [CrossRef]
14. Gardner, P.; Maffeis, S. Modelling dynamic web data. *Theor. Comput. Sci.* **2005**, *342*, 104–131. [CrossRef]
15. Sangiorgi, D. On the origins of bisimulation and coinduction. *ACM Trans. Program. Lang. Syst.* **2009**, *31*, 15:1–15:41. [CrossRef]
16. Ginzburg, A. *Algebraic Theory of Automata*, 1st ed.; Academic Press: Cambridge, MA, USA, 1968.

17. Milner, R. An Algebraic Definition of Simulation between Programs. In Proceedings of the 2nd International Joint Conference on Artificial Intelligence, London, UK, 1–3 September 1971; Cooper, D.C., Ed.; William Kaufmann: Pleasant Hill, CA, USA, 1971; pp. 481–489.
18. van Benthem, J. *Modal Logic and Classical Logic*; Bibliopolis: Asheville, NC, USA, 1985.
19. Forti, M.; Honsell, F. Set theory with free construction principles. *Ann. Della Sc. Norm. Super. Pisa Cl. Sci. 4E SÉRie* **1983**, *10*, 493–522.
20. Hinnion, R. Extensional quotients of structures and applications to the study of the axiom of extensionality. *Bull. Soci'Et'E Mathmatique Belg.* **1981**, *XXXIII*, 173–206.
21. Aman, B.; Ciobanu, G.; Koutny, M. Behavioural Equivalences over Migrating Processes with Timers. In Proceedings of the Formal Techniques for Distributed Systems—Joint 14th IFIP WG 6.1 International Conference, FMOODS 2012 and 32nd IFIP WG 6.1 International Conference, FORTE 2012, Stockholm, Sweden, 13–16 June 2012; Lecture Notes in Computer Science. Giese, H., Rosu, G., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7273, pp. 52–66. [CrossRef]
22. Clarke, E.M.; Grumberg, O.; Peled, D.A. *Model Checking*; MIT Press: Cambridge, MA, USA, 2001.
23. Milo, T.; Suciu, D. Index Structures for Path Expressions. In Proceedings of the Database Theory—ICDT '99, 7th International Conference, Jerusalem, Israel, 10–12 January 1999; Lecture Notes in Computer Science. Beeri, C., Buneman, P., Eds.; Springer: Berlin/Heidelberg, Germany, 1999; Volume 1540, pp. 277–295. [CrossRef]
24. Fan, W.; Li, J.; Wang, X.; Wu, Y. Query preserving graph compression. In Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, 20–24 May 2012; Candan, K.S., Chen, Y., Snodgrass, R.T., Gravano, L., Fuxman, A., Eds.; ACM: New York, NY, USA, 2012; pp. 157–168. [CrossRef]
25. Givan, R.; Dean, T.L.; Greig, M. Equivalence notions and model minimization in Markov decision processes. *Artif. Intell.* **2003**, *147*, 163–223. [CrossRef]
26. Kurtonina, N.; de Rijke, M. Expressiveness of Concept Expressions in First-Order Description Logics. *Artif. Intell.* **1999**, *107*, 303–333. [CrossRef]
27. Abriola, S.; Barceló, P.; Figueira, D.; Figueira, S. Bisimulations on Data Graphs. *J. Artif. Intell. Res.* **2018**, *61*, 171–213. [CrossRef]
28. Fagin, R.; Halpern, J.Y.; Moses, Y.; Vardi, M.Y. *Reasoning about Knowledge*; MIT Press: Cambridge, MA, USA, 1995. [CrossRef]
29. van Ditmarsch, H.; French, T.; Velázquez-Quesada, F.R.; Wáng, Y.N. Knowledge, awareness, and bisimulation. In Proceedings of the 14th Conference on Theoretical Aspects of Rationality and Knowledge (TARK 2013), Chennai, India, 7–9 January 2013; Schipper, B.C., Ed.; 2013.
30. Velázquez-Quesada, F.R. Bisimulation characterization and expressivity hierarchy of languages for epistemic awareness models. *J. Log. Comput.* **2018**, *28*, 1805–1832. [CrossRef]