Probabilistic and Prioritized Data Retrieval in the Linda Coordination Model ⋆

Mario Bravetti, Roberto Gorrieri, Roberto Lucchi, and Gianluigi Zavattaro

Dipartimento di Scienze dell'Informazione, Università degli Studi di Bologna,
Mura Anteo Zamboni 7, I-40127 Bologna, Italy.
e-mail:{bravetti,gorrieri,lucchi,zavattar}@cs.unibo.it

Abstract. Linda tuple spaces are flat and unstructured, in the sense that they do not allow for expressing preferences of tuples; for example, we could be interested in indicating tuples that should be returned more frequently w.r.t. other ones, or even tuples with a low relevance that should be taken under consideration only if there is no tuple with a higher importance. In this paper we investigate, in a process algebraic setting, how probabilities and priorities can be introduced in the Linda coordination model in order to support a more sophisticated data retrieval mechanism. As far as probabilities are concerned, we show that the Linda pattern-matching data retrieval makes it necessary to deal with weights instead of just pure probabilities, as instead can be done in standard process algebras. Regarding priorities, we present two possible ways for adding them to Linda; in the first one the order of priorities is statically fixed, in the second one it is dynamically instantiated when a data-retrieval operation is executed.

1 Introduction

The native Linda coordination model is based on a shared tuple-space which is an unstructured and flat bag of tuples. All tuples have the same importance and relevance inside the shared repository, in the sense that when several tuples match the template of a data-retrieval operation, one of them is selected non deterministically. In some applications, we may be interested in expressing more sophisticated policy for selecting the tuple to be returned, for example, according to some priority based access (one tuple should be returned only if no other tuples of higher priority are currently available) or a probabilistic selection (one tuple should be returned with a higher probability w.r.t. another one).

Consider, for example, the problem of coordinating the collaboration among Web-Services; in particular, consider the problem of discovering a Web-Service willing to offer a particular service. A tuple-space could be exploited in this scenario as a registry where the available Web-Services register the kind of services they intend to offer, while the clients access the tuple-space in order to discover

⋆ Work partially supported by MEFISTO Progetto "Metodi Formali per la Sicurezza e il Tempo".

the actual Web-Services availability. In the case more than one service is willing to offer the required service, according to the standard data-retrieval mechanism, one is chosen non deterministically. This is not satisfactory if we intend to distribute in a balanced way the workload, thus to avoid the overwhelming of requests towards one Web-Service while leaving other Services under-utilized.

As a second example, consider a master-worker application where the masters and the workers coordinate via a tuple-space: the masters produce job requests and store them inside the tuple-space, and the workers access the tuple-space to retrieve the description of the jobs to execute. We could assume that the jobs have different urgency levels, and that the workers must select a job for execution only if no jobs are currently registered with a higher priority. Another more sophisticated form of priority could be related to time constraints, e.g. assuming that a job must be executed before an expiration time, otherwise it can be left unexecuted. In this case, in order to limit the number of unexecuted jobs, each worker should select the job with the closest expiration time.

In the traditional Linda model these coordination patterns are rather difficult to be programmed. The reason is that the standard Linda data-retrieval coordination primitives (in and rd) access the tuple space on the basis of a *local* property, i.e., the conformance of the tuple w.r.t. the indicated template independently of the context in which the tuple is actually inserted. On the other hand, in the above examples *global* properties involving the entire group of matching tuples come into play. More precisely, in the examples the data-retrieval operation should select the tuple to be returned according to some function (either probabilistic or priority-based) that has all the matching tuples in its domain.

Extensions of Linda exist that support *global* operations. Consider, e.g., the $collect$ primitive of [9] (that permits to withdraw all the tuples satisfying the template) or the non blocking inp operation supported in some Linda system [5] (that returns one tuple matching the template, if available, or terminates indicating the absence of matching tuples). These primitives permit to program the coordination patterns described above, but in a rather unsatisfactory way.

For example, if we want to force a specific probabilistic distribution of the returned tuple, we could decorate each tuple adding (as an extra field) a value that quantifies the level of relevance of the tuple. When a data-retrieval operation is executed from an agent, this agent could $collect$ all the tuples satisfying the template, select the tuple according to the distribution of these values, and reintroduce the tuples in the space. Clearly, this pattern is not satisfactory because it requires to move from the tuple-space to the agent (and back) possibly huge quantities of tuples, and moreover this complex operation should be executed in a transactional manner, thus requiring consistent locks.

As far as the master-worker example is concerned, the priority-based coordination pattern that it requires could be programmed exploiting the inp operation. The level of priority could be associated to the tuples as an extra field. The workers that access the space could initially perform an inp taking into account the first level of priority, and passing to the subsequent levels only if no tuples are retrieved. This solution is satisfactory only if few levels of priority are considered, because it is necessary to explicitly access one level at a time. If the priorities are expressed in terms of an expiration time as described in the example above, this solution is clearly unfeasible because it requires a separate *inp* operation for each possible expiration time.

In light of these observations, we consider worth to investigate extensions of the Linda tuple-space coordination model with *probabilistic* as well as *prioritized* data retrieval. By probabilistic access we mean the possibility to specify the probabilistic distribution according to which the data to be returned are selected when a template is matched by more than one tuple. We observe that the most reasonable way for specifying this distribution is to associate weights to tuples indicating the absolute relevance of one tuple; when a data retrieval operation is performed, the weights of all the tuple matching the template are taken into account, and the tuple is probabilistically selected according to its relative weight w.r.t. the weights of the other tuples. Using this approach we can satisfactorily solve the problem of a balanced distribution of the workload of the Web-Services: each Web-Service indicates with a weight its current workload (the higher is the workload, the lower is the weight). When a client performs its discovery operation, a link to a Web-Service currently unloaded is more probably returned w.r.t. an overwhelmed one.

As far as the priority-based access is concerned, we investigate two possible scenarios; in the first one the order of the priorities is statically fixed; in the second one tuples are associated with a *symbolic* priority and the order relation among the symbolic priorities is defined from the processes at the time they execute the data-retrieval operations. Going back to the master-worker example, the first approach can easily solve the case in which there is a fixed hierarchy of urgency levels, simply by associating a different priority to each urgency level. On the other hand, in the case the job with the closest expiration time should be selected, we can proceed as follows: the expiration time of a job indicates its symbolic priority and, when a worker withdraws a job, it defines an order relation that privileges those with an expiration which is closer to the current time.

The paper is structured as follows. In Section 2 we present a formal description of the Linda coordination model that we use as the basis for our probabilistic and prioritized extensions. In Section 3 we discuss and formally introduce our probabilistic version of Linda, while in Section 4 we discuss and define the two extensions with a prioritized access to tuples. Finally, in Section 5 we draw some conclusive remark.

2 The Linda coordination model

The coordination primitives that we have in Linda are: $out(e)$, $in(t)$ and $rd(t)$. The output operation $out(e)$ inserts a tuple e in the tuple space (TS for short). Primitive $in(t)$ is the blocking input operation: when an occurrence of a tuple e matching with t (denoting a template) is found in the TS, it is removed from the TS and the primitive returns the tuple. The read primitive $rd(t)$ is the blocking read operation that, differently from $in(t)$, returns the matching tuple e without removing it from the TS.

Linda tuples are ordered and finite sequences of typed fields, while template are ordered and finite sequences of fields that can be either *actual* or *formal* (see [5]): a field is actual if it specifies a type and a value, whilst it is formal if the type only is given. For the sake of simplicity, in the formalization of Linda we are going to present fields are not typed.

Formally, let $Mess$, ranged over by m, m', ..., be a denumerable set of messages and Var, ranged over by x, y, ..., be the set of data variables. In the following, we use $\boldsymbol{x}$, $\boldsymbol{y}$, ..., to denote finite sequences $x_1; x_2; \ldots; x_n$ of variables.

Tuples, denoted by e, e', ..., are finite and ordered sequences of data fields, whilst templates, denoted by t, t', ..., are finite and ordered sequences of fields that can be either data or wildcards (used to match with any message).

Formally, tuples are defined as follows:

$$e = < \boldsymbol{d} >$$

where $\boldsymbol{d}$ is a term of the following grammar:

$\boldsymbol{d} ::= d \mid d; \boldsymbol{d}$
$d ::= m \mid x.$

The definition of template follows:

$$t = < \boldsymbol{dt} >$$

where $\boldsymbol{dt}$ is a term of the following grammar:

$\boldsymbol{dt} ::= dt \mid dt; \boldsymbol{dt}$
$dt ::= d \mid null.$

A *data field* d can be a message or a variable. The additional value *null* denotes the wildcard, whose meaning is the same of formal fields of Linda, i.e. it matches with any field value. In the following, the set $Tuple$ (resp. $Template$) denotes the set of tuples (resp. templates) containing no variable.

The matching rule between tuples and templates we consider is the classical one of Linda, whose definition is as follows.

Definition 1. ***Matching rule -*** *Let $e =< d_1; d_2; \ldots; d_n > \in Tuple$ be a tuple, $t =< dt_1; dt_2; \ldots; dt_m > \in Template$ be a template; we say that e matches t (denoted by $e \triangleright t$) if the following conditions hold:*

1. $m = n$.
2. $dt_i = d_i$ *or* $dt_i = null$, $1 \leq i \leq n$.

Condition 1. checks if e and t have the same arity, whilst 2. tests if each non-wildcard field of t is equal to the corresponding field of e.

Processes, denoted by P, Q, ..., are defined as follows:

$$
\begin{array}{lll}
P,\, Q,\, \ldots ::= & & \text{processes} \\
\quad \mathbf{0} & & \text{null process} \\
\quad \mid \ \mathit{out}\ (e).P & & \text{output} \\
\quad \mid \ \mathit{rd}\ t(\boldsymbol{x}).P & & \text{read} \\
\quad \mid \ \mathit{in}\ t(\boldsymbol{x}).P & & \text{input} \\
\quad \mid \ P \mid P & & \text{parallel composition} \\
\quad \mid \ !P & & \text{replication}
\end{array}
$$

A process can be a terminated program $\mathbf{0}$, a prefix form $\mu.P$, the parallel composition of two programs, or the replication of a program. The prefix μ can be one of the following coordination primitives: i) $\mathit{out}\ (e)$, that writes the tuple e in the TS; ii) $\mathit{rd}\ t(\boldsymbol{x})$, that given a template t reads a matching tuple e in the TS and stores the return value in $\boldsymbol{x}$; iii) $\mathit{in}\ t(\boldsymbol{x})$, that given a template t consumes a matching tuple e in the TS and stores the return value in $\boldsymbol{x}$. In both the $\mathit{rd}\ t(\boldsymbol{x})$ and $\mathit{in}\ t(\boldsymbol{x})$ operations $(\boldsymbol{x})$ is a binder for the variables in $\boldsymbol{x}$. The parallel composition $P \mid Q$ of two processes P and Q behaves as two processes running in parallel, whilst the replication operator $!P$ denotes the parallel composition of infinite copies of P.

In the following, $P[d/x]$ denotes the process that behaves as P in which all occurrences of x are replaced with d. We also use $P[\boldsymbol{d}/\boldsymbol{x}]$ to denote the process obtained by replacing in P all occurrences of variables in $\boldsymbol{x}$ with the corresponding value in $\boldsymbol{d}$, i.e. $P[d_1; d_2; \ldots; d_n/x_1; x_2; \ldots; x_n] = P[d_1/x_1][d_2/x_2] \ldots [d_n/x_n]$.

We say that a process is *well formed* if each prefix operation of kind $rd/in < \boldsymbol{dt} > (\boldsymbol{x})$ is such that the variables $\boldsymbol{x}$ and the data $\boldsymbol{dt}$ have the same arity. Notice that in the $\mathit{rd}\ t(\boldsymbol{x})$ and $\mathit{in}\ t(\boldsymbol{x})$ operations we use a notation which is different from the standard Linda notation: we explicitly indicate in $(\boldsymbol{x})$ the variables that will be bound to the actual fields of the matching tuple, while in the standard Linda notation these variables are part of the template. Observe that the two notations are equivalent up to the fact that our notation introduces variables also in association to the formal fields of the template. We also say that a process is *closed* if it has no free variable. In the following, we consider only processes that are closed and well formed; $\mathit{Process}$ denotes the set of such processes.

Let DSpace, ranged over by DS, DS', ..., be the set of possible configurations of the TS, that is $\mathit{DSpace} = \mathcal{M}_{fin}(\mathit{Tuple})$, where $\mathcal{M}_{fin}(S)$ denotes the set of all the possible finite multisets on S. In the following, we use $DS(e)$ to denote the number of occurrences of e within $DS \in \mathit{DSpace}$.The set $\mathit{System} = \{[P, DS] \mid P \in \mathit{Process}, DS \in \mathit{DSpace}\}$, ranged over by s, s', ..., denotes the possible configurations of systems.

The semantics we use to describe processes interacting via Linda primitives is defined in terms of a transition system $(\mathit{System}, \longrightarrow)$, where $\rightarrow \subseteq \mathit{System} \times \mathit{System}$. More precisely, $\longrightarrow$ is the minimal relation satisfying the axioms and rules of Table 1 (symmetric rule of (4) is omitted). $(s, s') \in \longrightarrow$ (also denoted by $s \longrightarrow s'$) means that a system s can evolve (performing a single action) in the system s'.

$$(1)\quad [out\ (e).P, DS] \longrightarrow [P, DS \oplus e]$$

$$(2)\quad \frac{\exists e \in DS : e \triangleright t}{[in\ t(\boldsymbol{x}).P, DS] \longrightarrow [P[e/\boldsymbol{x}], DS - e]}$$

$$(3)\quad \frac{\exists e \in DS : e \triangleright t}{[rd\ t(\boldsymbol{x}).P, DS] \longrightarrow [P[e/\boldsymbol{x}], DS]}$$

$$(4)\quad \frac{[P, DS] \longrightarrow [P', DS']}{[P \mid Q, DS] \longrightarrow [P' \mid Q, DS']}$$

$$(5)\quad \frac{[P, DS] \longrightarrow [P', DS']}{[!P, DS] \longrightarrow [P' \mid !P, DS']}$$

Table 1. Semantics of Linda

Axiom (1) describes the output operation that produces a new occurrence of the tuple e in the shared space DS ($DS \oplus e$ denotes the multiset obtained by DS increasing by 1 the number of occurrences of e). Rules (2) and (3) describe the in and the rd operations, respectively: if a matching e tuple is currently available in the space, it is returned at the process invoking the operation and, in the case of in, it is also removed from the space ($DS - e$ denotes the removal of an occurrence of e from the multiset DS). Rule (4) represents a local computation of processes, whilst (5) the replication operator that produces a new instance of the process and copies itself.

3 Adding probabilistic data-retrieval to Linda

In this section we extend the Linda language by introducing probabilities for retrieval of tuples stored in the TS. We start by discussing the probabilistic model that we will adopt (Section 3.1) and then we present the new extended Linda primitives: their syntax (Section 3.2) and their semantics (Section 3.3).

3.1 Probabilistic model

In Linda expressing a probabilistic choice among entities reacting to a given communication request (e.g. tuples matching a “rd” or “in” request) requires a much more complex mechanism w.r.t. languages employing channel-based communication (like, e.g., those representable by standard process algebras). This is due to the greater complexity of Linda matching-based communication mechanism w.r.t. such a simpler form of communication. If we had channel-based communication then we could just consider probability distributions (i.e. functions assigning probabilities that sum up to 1 to elements of a given domain) over the messages $a(\boldsymbol{d})$ actually available on the channel type a; when a receive operation is performed on the channel of type a, the channel would “react” to the request by simply choosing a message $a(\boldsymbol{d})$ for some d according to such a probability distribution (this is what would happen by adopting either the “reactive model” of probability [12] or the “simple model” of [10]). When we consider the Linda matching-based communication mechanism, we loose the separation above between the channel type (which decides the set of entities involved in the communication) and the datum d that is read. Since the set of matching tuples $< \boldsymbol{d} >$ is now established from a template t on data that is chosen by the “rd” or “in” operation, it is unavoidable to deal with the situation in which the set of matching tuples is a proper subset of the domain of a probability distribution: in this case a re-normalization of “probability” values must be done in order to have them summing up to 1 (this is the same situation that arises for the restriction operator in generative models [12]). Note that the only way to avoid this would be to have an individual probability distribution for each datum $< \boldsymbol{d} >$ that is present in the shared space (over the several instances of such datum), i.e. by treating each different datum $< \boldsymbol{d} >$ in the same way as a channel type a in channel-based communication. However, since in this case the channel type would coincide with the datum that is read from the tuple space, reading (or consuming) different tuples having the same channel type (i.e. different instances of the same datum $< \boldsymbol{d} >$) would have the same observable effect on the system, hence probability distributions would be useless.

As a consequence of this remark, since when the shared space is accessed the probabilities on matching tuples must be determined by using re-normalization (on the basis of the “selecting power” of the template in the “rd” or “in” operation), it is natural to express probability statically associated to tuples in the space by means of *weights* [11]. A weight is a positive real number which is associated to an entity that can be involved in a probabilistic choice: the actual probability that the entity assumes in the choice is determined from the associated weight depending on the context, i.e. from the weights of the other entities involved in the choice, by re-normalization.

Example 1. We indicate the weight w of a tuple associating the notation $[w]$ to the tuple itself. Let us suppose that the tuple space contains three tuples $< m_1, m_2 > [w]$, $< m_1, m_2' > [w']$ and $< m_1', m_2'' > [w'']$, then the following happens. If the operation $rd \ < null, null > (x_1, x_2).P$ is performed, the variables x_1, x_2 are bound: to m_1, m_2 with probability $w/(w + w' + w'')$, to m_1, m_2' with probability $w'/(w+w'+w'')$, and to m_1', m_2'' with probability $w''/(w+w'+w'')$. If the operation $rd \ < m_1, null > (x_1, x_2).P$ is performed, the variables x_1, x_2 are bound: to m_1, m_2 with probability $w/(w + w')$ and to m_1, m_2' with probability $w'/(w + w')$. If the operation $rd \ < m_1, m_2 > (x_1, x_2).P$ is performed, the variables x_1, x_2 are bound: to m_1, m_2 with probability $w/w = 1$.

Moreover note that since the structure of the shared space is highly dynamic and tuples are introduced individually in the space, expressing weights associated to tuples seems to be preferable w.r.t. expressing a single probabilistic distribution over all tuples (generative approach of [12]) which is to be updated by re-normalization to value 1 every time a tuple is added or removed. Therefore, due to the inherent re-normalization behavior of Linda, and to the observations we have made, we adopt, like in [2, 1], the approach above based on weights.

3.2 Syntax

Formally, let $Weight$, ranged over by w, w', ..., be the set of weights, i.e. positive (non-zero) real numbers. Tuples are now decorated with an attribute indicating the associated weight (representing their "appealing degree"), whilst templates have the classical structure as in Linda.

Tuples, denoted by e, e', ..., are now defined as follows:

$$e = < \boldsymbol{d} > [w]$$

where $w \in Weight$ and $\boldsymbol{d}$ is a sequence of data fields (see Section 2) d that are defined by the following grammar:

$d ::= m \mid w \mid x.$

A *data field* d can be now a message, a weight, or a variable. We also define $\tilde{\ }$ as the function that, given a tuple e, returns its sequence of data fields (e.g. if $e = < \boldsymbol{d} > [w]$ then $\tilde{e} = \boldsymbol{d}$), and a function W that, given a tuple, returns its weight (e.g., $W(< \boldsymbol{d} > [w]) = w$).

Similarly to Section 2, in the following we make use of $Tuple$ and $Template$ to denote sets of tuples and template. In general here and in the following sections we will assume them to take into account the current definition of tuples and templates. The same holds for the matching rule between tuples and templates, denoted by "$\triangleright$", which is the classical one of Linda defined in Section 2 (the matching evaluation does not take into account the weight attributes associated to tuples).

The extended version of the Linda primitives has the following meaning:

- *out*(e), where $e \in Tuple$ is the output operation; given a tuple e, where $e = < \boldsymbol{d} > [w]$, it writes it into the DS.
- *in*(t), where $t \in Template$; if some tuple e matching with template t is available in the DS, the execution of in causes the removal of one of such tuples e from the space and returns $\tilde{e}$. The probability of removing a particular tuple $e = < \boldsymbol{d} > [w] \in DS$ with e that matches t is the ratio of w to the sum of the weights w' in the tuples $e' = < \boldsymbol{d'} > [w']$ in the DS such that e' matches with t (taking into account multiple occurrences of tuples).
- *rd*(t), where $t \in Template$; if some tuple e matching with template t is available in the DS, one of such tuples is read and the returned value is $\tilde{e}$. The probability of reading a particular tuple e with e that matches t is

evaluated as in the input case.
Note that this means that the probability of reading a particular matching sequence of data fields $< \boldsymbol{d} >$ contained in the DS is the ratio of the sum of weights w associated with the several instances of $< \boldsymbol{d} >$ contained in the DS, to the sum of the weights of the tuples e' in the DS matching with t.

It is worth noting that the probabilistic access to the tuples is at the level of the subspace that the agent can access using a specific template. More precisely, the probability distribution depends on the weights of all matching tuples stored in the DS.

3.3 Semantics

In this section we introduce the semantics of systems interacting via the probabilistic model of Linda. In particular, we describe how we replace -in the data retrieval operations- the standard non-deterministic choice of a tuple among the matching ones in the TS, with a probabilistic choice exploiting weights.

Let $Prob = \{\rho \mid \rho : System \longrightarrow [0,1] \wedge supp(\rho)$ is finite $\wedge \sum_{s \in System} \rho(s) = 1\}$, where $supp(\rho) = \{s \mid \rho(s) > 0\}$, be the set of probability distributions on configurations. The semantics we use to describe systems is defined in terms of probabilistic transition systems $(System, Prob, \longrightarrow)$, where $\rightarrow \subseteq System \times Prob$. More precisely, $\longrightarrow$ is the minimal relation satisfying the axioms and rules of Table 2. $(s, \rho) \in \longrightarrow$ (also denoted by $s \longrightarrow \rho$) means that a system s can reach a generic configuration s' with a probability equal to $\rho(s')$. Note that, several probability distributions may be performable from the same state s, i.e. it may be that $s \longrightarrow \rho$ for several different ρ. This means that (like in the simple model of [10]) whenever the system is in state s, first a non-deterministic choice is performed which decides which of the several probability distributions ρ must be considered, then the next configuration is probabilistically determined by the chosen distribution ρ. Note that the non-deterministic choice may, e.g., arise from several concurrent rd operations which probabilistically retrieve data from the tuple-space. We use $s \longrightarrow s'$ to denote $s \longrightarrow \rho$, with ρ the trivial distribution which gives probability 1 to s' and probability 0 to all other configurations.

Table 3 defines: (i) the probability distributions $\rho^{p}_{in\ t(\boldsymbol{x}).P,DS}$ and $\rho^{p}_{rd\ t(\boldsymbol{x}).P,DS}$ used for in and rd operations, respectively; (ii) the operator $\rho|Q$ that, given ρ, computes a new probability distribution that accounts for composition with "Q". It is worth noting that $\rho^{p}_{in\ t(\boldsymbol{x}).P,DS}$ and $\rho^{p}_{rd\ t(\boldsymbol{x}).P,DS}$ are defined only for $t \in Template$ and $DS \in DSpace$ such that there exists $e \in DS\!: e \rhd t$ (that is the condition reported in axioms (2) and (3)).

Axiom (1) describes the output of the tuple e, after the execution an occurrence of e is added to the shared space DS and the process continues with P. Axiom (2) describes the behaviour of in operations; if a tuple e matching with template t is available in the DS, the in execution produces the removal from the space of e and then the process behaves as $P[\tilde{e}/\boldsymbol{x}]$. The probability of reaching a configuration where a matching tuple e contained in the DS is removed is the ratio of the total weight of the several instances of e in the DS, to the sum of the

$$(1)\quad [out\ (e).P, DS] \longrightarrow [P, DS \oplus e]$$

$$(2)\quad \frac{\exists e \in DS : e \triangleright t}{[in\ t(\boldsymbol{x}).P, DS] \longrightarrow \rho^{p}_{in\ t(\boldsymbol{x}).P,DS}}$$

$$(3)\quad \frac{\exists e \in DS : e \triangleright t}{[rd\ t(\boldsymbol{x}).P, DS] \longrightarrow \rho^{p}_{rd\ t(\boldsymbol{x}).P,DS}}$$

$$(4)\quad \frac{[P, DS] \longrightarrow \rho}{[P \mid Q, DS] \longrightarrow \rho|Q}$$

$$(5)\quad \frac{[P, DS] \longrightarrow \rho}{[!P, DS] \longrightarrow \rho|!P}$$

Table 2. Semantics of Linda with probabilistic access to tuples

$$\rho^{p}_{in\ t(\boldsymbol{x}).P,DS}(s) = \begin{cases} \dfrac{W(e) \cdot DS(e)}{\sum_{e' \in DS : e' \triangleright t} W(e') \cdot DS(e')} & if\ \ s = [P[\tilde{e}/\boldsymbol{x}], DS - e] \\ & \quad with\, e \in DS\ \wedge\ e \triangleright t \\ 0 & o.w. \end{cases}$$

$$\rho^{p}_{rd\ t(\boldsymbol{x}).P,DS}(s) = \begin{cases} \dfrac{\sum_{e' \in DS : e' \triangleright t \wedge P[\tilde{e'}/\boldsymbol{x}] = P[\tilde{e}/\boldsymbol{x}]} W(e') \cdot DS(e')}{\sum_{e' \in DS : e' \triangleright t} W(e') \cdot DS(e')} & if\ \ s = [P[\tilde{e}/\boldsymbol{x}], DS] \\ & \quad with\, e \in DS\ \wedge\ e \triangleright t \\ 0 & o.w. \end{cases}$$

$$\rho|Q(s) = \begin{cases} \rho([P', DS]) & if\ \ s = [P' \mid Q, DS], \\ & \quad P' \in Process \wedge\ DS \in DSpace. \\ 0 & o.w. \end{cases}$$

Table 3. Probability distributions

total weights of the several instances of the matching tuples currently available in the DS. In this way, the probability to reach a system configuration takes into account the multiple ways of removing e due to the several occurrences of e in the DS. The axiom (3) describes rd operations; if a tuple e matching with template t is available in the DS, then the process behaves as $P[\tilde{e}/\boldsymbol{x}]$. Differently from the previous axiom, rd operations do not modify the tuple space, i.e. reached states do not change the configuration of DS, therefore they are simply differentiated by the continuation $P[\tilde{e}/\boldsymbol{x}]$ of the reading process. For example, let us consider two different tuples e' and e'' that contain the same value in some of their fields. If the reading process considers only the common fields, we have that $P[\tilde{e'}/\boldsymbol{x}] = P[\tilde{e''}/\boldsymbol{x}]$, thus it is not possible to discriminate the selection of the two different tuples. Therefore, the probability of reaching a configuration s that is obtained by reading a tuple e matching with t in the DS (yielding value e) is the ratio of the sum of the total weights associated to the several instances of tuples e' matching with t in the DS such that the continuation of the reading process obtained by reading tuple e' is the same as the one obtained by reading e, to the sum of the total weights of the several instances of the matching tuples currently available in the DS. Rule (4) describes the behaviour of the parallel composition of processes (the symmetric rule is omitted): if configurations reachable from $[P, DS]$ are described by the probability distribution ρ, and P performs an action in the system $[P \mid Q, DS]$ (the process that proceeds between P and Q is non-deterministically selected), then the reachable configurations are of the form $[P' \mid Q, DS']$, for some $P' \in Process$ and $DS' \in DSpace$. The probability values of such configurations do not depend on Q (that is "inactive") and are equal to $\rho([P', DS'])$. Finally, rule (5) describes the behaviour of process replication operator: $!P$ behaves as an unbounded parallel composition of the process P.

4 Adding prioritized data-retrieval to Linda

Priorities on tuples represent an absolute preference of the currently available tuples in the shared space. More precisely, if a process performing a rd/in operation receives as the return value a tuple e, there is no currently available matching tuple e' in TS such that its priority level is greater than the one of e. In Section 4.1 we present a model with *static priorities*: priorities on tuples are set by the output operations and the data-retrieval operations return a matching tuple with the highest priority among the available ones. In Section 4.2 we propose another solution which supports *dynamic priorities*: tuples are partitioned by associating a partition name with each tuple (set by the output operations) and processes performing rd/in operations can express a priority level to be dynamically assigned to partitions for the data-retrieval.

4.1 Static priorities

Following the approach used to introduce weights, tuples are now further extended by adding, similarly as in [2, 1], an additional attribute indicating their priority. Templates keep the classical structure as well as the classical Linda matching rule given in Definition 1.

Formally, let $Priority$, ranged over by $l, l', \ldots$, be the set of possible priorities, i.e. positive (non-zero) natural numbers.

Tuples are now defined as follows:

$$e =< \boldsymbol{d} > [w, l]$$

where $w \in Weight$, $l \in Priority$ and $\boldsymbol{d}$ is a sequence of data fields d that are defined by the following grammar:

$d ::= m \mid w \mid l \mid x.$

A data field d now can be a message, a weight, a priority level, or a variable.

In the following, we denote with PL the function that, given a tuple, returns its priority level (e.g., if $e =< \boldsymbol{d} > [w, l]$ then $PL(e) = l$), and with DS_l the partition of DS determined by selecting all the tuples e with priority level l, i.e. the multiset contained in DS such that for any $e \in Tuple$ if $e \in DS$ and $PL(e) = l$ then $DS_l(e) = DS(e)$, otherwise $DS_l(e) = 0$. We also define $L(DS, t)$ as the function that, given a space DS and a template t, returns the highest priority level of tuples matching with t,i.e. $L(DS, t) = \max\{l \mid \exists e \in DS_l : e \triangleright t\}$. Note that the function $L(DS, t)$ is defined only in the case DS contains at least one matching tuple within DS and, since we consider only DS that are finite multisets of $Tuple$, it can be computed for any $DS \in DSpace$.

The semantics of the Linda model with priority and weights on tuples is obtained from Table 2, by replacing $\rho^p_{rd\ t(\boldsymbol{x}).P,DS}$ and $\rho^p_{in\ t(\boldsymbol{x}).P,DS}$ with $\rho^{p,l}_{rd\ t(\boldsymbol{x}).P,DS}$ and $\rho^{p,l}_{in\ t(\boldsymbol{x}).P,DS}$, respectively, which are defined in Table 4. The definition of the new probability distributions in Table 4 makes use of the probability distributions previously defined in Table 3 for the pure probabilistic case. Informally, the idea is that the search space is restricted to the partition of DS which has the greatest priority level and contains a matching tuple. On this subspace, the probabilistic data-retrieval is governed by weights of the matching tuples it contains. More precisely, $\rho^{p,l}_{in\ t(\boldsymbol{x}).P,DS}([P', DS'])$ is defined by using the corresponding probabilistic version applied on the restricted space, i.e. $\rho^p_{in\ t(\boldsymbol{x}).P,DS_{L(DS,t)}}([P', DS'_{L(DS,t)}])$. This "reduction" is applied only if $DS - DS_{L(DS,t)} = DS' - DS'_{L(DS,t)}$ (where "$-$" is the usual multiset difference operator), that is if the partitions of the space not involved in the restrictions of DS and DS' used by ρ^p do not change. The same kind of "reduction" and condition have been used to define the probability distribution for rd operations.

4.2 Dynamic name-based priorities

Priority attributes on tuples introduced in the previous section logically partition the tuple space: each partition is identified by a priority level and contains all tuples with that specific priority level, i.e. $DS = \sum_{l \in Priority} DS_l$, for any $DS \in DSpace$.

$$\rho^{p,l}_{in\ t(\boldsymbol{x}).P,DS}([P',DS']) = \begin{cases} \rho^{p}_{in\ t(\boldsymbol{x}).P,DS_{L(DS,t)}}([P',DS'_{L(DS,t)}]) & if\ DS - DS_{L(DS,t)} = \\ & \quad DS' - DS'_{L(DS,t)} \\ 0 & o.w. \end{cases}$$

$$\rho^{p,l}_{rd\ t(\boldsymbol{x}).P,DS}([P',DS']) = \begin{cases} \rho^{p}_{rd\ t(\boldsymbol{x}).P,DS_{L(DS,t)}}([P',DS'_{L(DS,t)}]) & if\ DS - DS_{L(DS,t)} = \\ & \quad DS' - DS'_{L(DS,t)} \\ 0 & o.w. \end{cases}$$

Table 4. Probability distributions with priority

In this section we discuss how to manage dynamically priorities by providing, at the process that is willing to perform a rd/in operation, a manner to express in which partition of the tuple space to search the matching tuple (by associating an absolute preference to partitions), thus providing a way to restrict the search space to a subset of the tuple space.

Tuples have now an attribute weight and another one that is a "key": the name used to identify the partition. In order to express in which partitions to search matching tuples and with which priority, templates are decorated by adding a partial function that maps keys on priority levels. In this way, producers of tuples do not assign an absolute preference on the tuples, that is dynamically described by the processes performing a in/rd operation.

Formally, let Key, ranged over by k, k', ..., be the set of keys, and $KL = \{f \mid f : Key \hookrightarrow Priority\}$, ranged over by f, f', ..., be the set of partial functions mapping keys to priority levels; we denote with $Dom(f)$ the domain of the function f.

The definition of tuple is as follows:

$$e = < \boldsymbol{d} > [w, k]$$

where $w \in Weight$, $k \in Key$ and $\boldsymbol{d}$ is a sequence of data fields d that can be set to a message, a key, a weight, or a variable. In the following, we use $PK(e)$ to denote the key associated to the tuple e, i.e. $PK(< \boldsymbol{d} > [w,k]) = k$.

The structure of templates is the following:

$$t = < \boldsymbol{dt} > [f]$$

where $f \in KL$ and $\boldsymbol{dt}$ is a sequence of data fields that, in addition to those ones used by tuples, can also be set to wildcard value.

We also define $DS_{\{k_1,k_2,\ldots,k_n\}}$ as the function that given a $DS \in DSpace$ and a set of keys $\{k_1, k_2, \ldots, k_n \mid k_i \in Key, 1 \leq i \leq n\}$ returns the multiset containing all the tuples in DS associated with one of the keys in the given set, i.e. for any $e \in Tuple$ if $e \in DS$ and $PK(e) = k_i$ for some $1 \leq i \leq n$ then $DS_{\{k_1,k_2,\ldots,k_n\}}(e) = DS(e)$, otherwise $DS_{\{k_1,k_2,\ldots,k_n\}}(e) = 0$.

Finally, we define the functions $L(DS, t, f)$ and $K(DS, t, f)$ that, given a configuration of the tuple space $DS \in DSpace$, a template $t \in Template$ and a function $f \in KL$, return the highest priority level and the set of keys with the highest priority level of the available matching tuples in DS, respectively. They are defined as it follows:

$$L(DS, t, f) = \max\{f(k) \mid k \in Dom(f) \wedge \exists e \in DS_{\{k\}} : e \triangleright t\}$$

$$K(DS, t, f) = \{k \mid k \in Dom(f) \wedge f(k) = L(DS, t, f)\}$$

Note that the functions L and K are defined only for DS, t and f such that there exists at least one matching tuple $e \in DS$ having an associated key k contained in the domain of f.

The semantics of the model with dynamic priority and weights on tuples is obtained by using rules (1), (4) and (5) of Table 2, rules (2') and (3') of Table 5 and the probability distributions reported in Table 6.

$$(2')\ \frac{\exists e \in DS : e \triangleright t \quad PK(e) \in Dom(f)}{[in\ <\boldsymbol{dt}>[f](\boldsymbol{x}).P, DS] \longrightarrow \rho^{p,dl}_{in\ <\boldsymbol{dt}>[f](\boldsymbol{x}).P,DS}}$$

$$(3')\ \frac{\exists e \in DS : e \triangleright t \quad PK(e) \in Dom(f)}{[rd\ <\boldsymbol{dt}>[f](\boldsymbol{x}).P, DS] \longrightarrow \rho^{p,dl}_{rd\ <\boldsymbol{dt}>[f](\boldsymbol{x}).P,DS}}$$

Table 5. *rd* and *in* semantics with dynamic priority

Differently from *rd* and *in* operations with priority proposed in the previous section, the corresponding ones with dynamic priority can be performed only if a tuple with one of the specified priority levels is found in the space. Therefore, (2') and (3') differ from (2) and (3) of Table 2 because we need to test not only the presence in DS of a matching tuple, but also that it has an associated symbol k that is in the domain of f (indicated by the template). The definition of the probability distributions of Table 6 follows the same idea used in those ones for static priority: we exploit the definition of probability distributions ρ^p by restricting the space DS to that one containing only tuples having a key corresponding to the highest priority level containing matching tuples.

$$\rho^{p,dl}_{in<\boldsymbol{dt}>[f](\boldsymbol{x}).P,DS}([P',DS']) = \begin{cases} \rho^{p}_{in<\boldsymbol{dt}>[f](\boldsymbol{x}).P,DS_{K(DS,t,f)}}([P',DS'_{K(DS,t,f)}]) & if\ DS - DS_{K(DS,t,f)} = \\ & DS' - DS'_{K(DS,t,f)} \\ 0 & o.w. \end{cases}$$

$$\rho^{p,dl}_{rd<\boldsymbol{dt}>[f](\boldsymbol{x}).P,DS}([P',DS']) = \begin{cases} \rho^{p}_{rd<\boldsymbol{dt}>[f](\boldsymbol{x}).P,DS_{K(DS,t,f)}}([P',DS'_{K(DS,t,f)}]) & if\ DS - DS_{K(DS,t,f)} = \\ & DS' - DS'_{K(DS,t,f)} \\ 0 & o.w. \end{cases}$$

Table 6. Probability distributions for dynamic priorities

5 Conclusion

In this paper we have introduced, in a process algebraic setting, probabilities and priorities on the the data-retrieval mechanisms of Linda. We have technically motivated the use of weights on tuples -instead of classic probability distribution- to express probabilities, and discussed two possible forms of priority mechanisms on tuples, one using static priorities and another one using dynamic priorities defined by the process when they execute data-retrieval operations. To the best of our knowledge, the unique paper that addresses probabilities in the context of Linda-like languages is Probabilistic Klaim [8]. Klaim [7] is a distributed mobile version of Linda where processes may migrate among nodes in a net; in Probabilistic Klaim probabilities are used to describe probabilistic forms of mobility.

In [3] we describe an implementation of a Linda-like repository that exploits the probabilistic data-retrieval described in this paper. This repository is intended to be used to distribute the workload among collaborating Web Services, as briefly described in the Introduction section. More precisely, we describe how to implement a registry service that supports registration, deregistration, discovery and update of Web Services descriptions. In particular, the update operation allows for run-time modification of the weight associated to a tuple.

As future work, a natural extension of the model with dynamic priorities is to provide a manner for limiting the scope of the symbolic priorities; in other words, we want to ensure that the data associated to a specific symbolic priority can be accessed only from a restricted group of processes. In terms of security properties, this corresponds to a form of secure group communication. Linda is not expressive enough to provide security solutions and, more precisely, access control mechanisms on the tuple space. Therefore, we need to model probabilities and priorities on a more sophisticated Linda-like language supporting these features (e.g., [4, 13, 7, 6]). In particular, we intend to investigate on a possible solution that exploits the access control mechanism provided by `SecSpaces` [4] because it is mainly based on partitions which shares common features with the notion of symbolic priorities.

References

1. M. Bravetti. *Specification and Analysis of Stochastic Real-Time Systems.* PhD thesis, Dottorato di Ricerca in Informatica. Università di Bologna, Padova, Venezia, February 2002. Available at `http://www.cs.unibo.it/~bravetti/`.
2. M. Bravetti and M. Bernardo. Compositional asymmetric cooperations for process algebras with probabilities, priorities, and time. In *Proc. of the 1st Int. Workshop on Models for Time-Critical Systems, MTCS 2000,* State College (PA), volume 39(3) of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2000.
3. Mario Bravetti, Roberto Gorrieri, Roberto Lucchi, and Gianluigi Zavattaro. Web Services for E-commerce: guaranteeing security access and quality of service. In *Proc. of ACM Symposium on Applied Computing (SAC'04)*. ACM Press, 2004.
4. Nadia Busi, Roberto Gorrieri, Roberto Lucchi, and Gianluigi Zavattaro. Secspaces: a data-driven coordination model for environments open to untrusted agents. In *1st International Workshop on Foundations of Coordination Languages and Software Architectures*, volume 68.3 of *ENTCS*, 2002.
5. Scientific Computing Associates. *Linda: User's guide and reference manual.* Scientific Computing Associates, 1995.
6. N. Minsky, Y. Minsky, and V. Ungureanu. Safe Tuplespace-Based Coordination in Multi Agent Systems. *Journal of Applied Artificial Intelligence*, 15(1), 2001.
7. R. De Nicola, G. Ferrari, and R. Pugliese. KLAIM: A Kernel Language for Agents Interaction and Mobility. *IEEE Transactions on Software Engineering*, 24(5):315–330, May 1998. Special Issue: Mobility and Network Aware Computing.
8. A. Di Pierro, C. Hankin, and H. Wiklicky. Probabilistic Klaim. In *Proc. of 7th International Conference on Coordination Models and Languages (Coordination 04)*, LNCS. Springer Verlag, 2004.
9. A. Rowstron and A. Wood. Solving the Linda multiple `rd` problem using the `copy-collect` primitive. *Science of Computer Programming*, 31(2-3):335–358, 1998.
10. R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1995.
11. C.M.N. Tofts. Processes with probabilities, priority and time. *Formal Aspects of Computing*, 6:534–564, 1994.
12. R.J. van Glabbeek, S.A. Smolka, and B. Steffen. Reactive, Generative and Stratified Models of Probabilistic Processes. *Information and Computation*, 121:59–80, 1995.
13. Jan Vitek, Ciarán Bryce, and Manuel Oriol. Coordinating Processes with Secure Spaces. *Science of Computer Programming*, 46:163–193, 2003.