ALMA MATER STUDIORUM
UNIVERSITÀ DEGLI STUDI DI BOLOGNA

Seconda Facoltà di Ingegneria
Corso di Laurea Specialistica in Ingegneria Informatica

SUPPORTO A REGOLE CHIMICO-SEMANTICHE PER LA COORDINAZIONE DI PERVASIVE SERVICE ECOSYSTEMS

Elaborata nel corso di:
Linguaggi e Modelli Computazionali LS

Tesi di Laurea di:
MATTEO DESANTI

Relatore:
Prof. MIRKO VIROLI

Correlatori:
Ing. ELENA NARDINI
Ing. DANILO PIANINI

ANNO ACCADEMICO 2010-2011
SESSIONE II

PAROLE CHIAVE

SAPERE Project
Chemical-Semantic Coordination
Service Ecosystems

A mio padre,
il mio primo insegnante

Indice

Introduzione

Il mondo in cui viviamo è ogni giorno di piú permeato e invaso da tecnologie informatiche in crescente evoluzione. Basti vedere lo sviluppo che si è avuto nel campo mobile dove dispositivi pensati per eseguire solo pochi semplici compiti sono, negli ultimi anni, sempre piú soppiantati da devices dalle altissime prestazioni, per le dimensioni ridotte che presentano, e dalle grandi capacità di connettività con il mondo circostante. Non solo i dispositivi sono sempre piú potenti ma spesso anche dotati di sensori per determinare informazioni quali la loro posizione o apprendere parte delle informazioni sulla realtà che li circonda. Tali funzionalità si rivelano utili anche in abiti diversi da quello mobile quali la *domotica* o l'*automotive* nel quale, per esempio, vediamo l'ascesa di auto dotate di tutti i confort tecnologici, dal telefono integrato agli antifurto satellitari, dai navigatori agli innovativi park assistant. L'alta connettività che caratterizza tutte le nuove tecnologie fà sì che le persone richiedano un grado di utilizzo delle reti e dei servizi che esse offrono via via maggiore e il Web risponde a tali necessità evolvendosi con la crescita di applicazioni e servizi caratterizzati da un continuo incremento del livello di interazione con l'utente.

Questo incalzante espandersi della tecnologia ha ben presto trasformato noi, suoi usufruitori, in produttori e consumatori di grandissime quantità di informazioni, molto spesso senza rendercene conto. Ci stiamo sempre piú spostando verso una realtà descrivibile secondo la visione del *pervasive* o *ubiquitous computing*, la quale vede l'interazione uomo-macchina e l'elaborazione delle informazioni decentralizzarsi dal singolo device (il personal computer) per integrarsi negli oggetti e nelle attività di tutti i giorni. Uno scenario di questo tipo offre indubbiamente ampi spazi di manovra per la nascita di sistemi innovativi che si basano su infrastrutture decentralizzate per la creazione di servizi pervasivi general-purpose. Questi sistemi possono includere le classi tradizionali di servizi arricchite con nuove funzionalità

quali la capacità di adattarsi dinamicamente e autonomamente al contesto in cui si trovano, si pensi per esempio a servizi di visualizzazione pubblicitaria su display pubblici che basano i loro contenuti sulla base delle preferenze di un utente posto vicino a essi, conservate all'interno del suo PDA o smartphone. Chiaramente si richiede un approccio completamente nuovo dovendo affrontare problematiche quali la coordinazione e interazione tra gli agenti e la distribuzione spaziale dei componenti del sistema. A ciò si aggiungano tutte quelle caratteristiche generalmente richieste che rendono i sistemi efficienti quali la robustezza a malfunzionamenti parziali, capacità di self-management (autogestione) e di self-optimisation (auto-ottimizzazione) che in un unica parola possiamo racchiudere sotto il concetto di *self-organisation* (auto-organizzazione).

Gli studi degli ultimi anni in tali ambiti hanno portato la ricerca a prendere sempre piú ispirazione per risolvere questa classe di problemi dal mondo naturale che ci circonda. Astrazioni naturali di vario tipo sono effettivamente caratterizzate da aspetti che rappresentano i requisiti richiesti negli scenari suddetti: *(i)* il concetto di spazialità dell'ambiente che interessa anche i meccanismi di coordinazione delle attività dei componenti che hanno una specifica località spaziale; *(ii)* la capacità di facilitare interazioni spontanee tra i componenti senza che sia richiesta agli stessi una conoscenza a priori reciproca; *(iii)* la flessibilità di tollerare l'evoluzione della struttura e la diversità con il passare del tempo mantenendo una maniera uniforme per modellare i propri componenti e la loro coordinazione [5]. Nonostante i sistemi naturali possano essere categorizzati in varie classi (e.g. fisici, chimici, biologici, o sociali) mettono tutti in luce una realtà costituita da un substrato ambientale spazialmente localizzato in cui individui autonomi di varie tipologie si incontrano, interagiscono , competono e combinano tra di loro (in una parola *coordinano*) sulla base di alcune semplici leggi naturali. La tesi in oggetto baserà il suo studio su un particolare paradigma di ispirazione chimico-biologica per modellare un infrastruttura in grado di permettere la coordinazione all'interno di un ecosistema di servizi pervasivi dotata dei requisiti di cui sopra.

Il lavoro eseguito nel corso della tesi si colloca nell'ambito del progetto *SAPERE* [2], acronimo di Self Aware Pervasive Services Ecosystems, promosso dall'iniziativa *STREP* (Specific Targeted Reserch Projects) e finanziato nell'ambito del programma FP7 dell'Unione Europea con l'obiettivo di creare una innovativa infrastruttura nature-inspired per supportare il deployment

e la coordinazione di servizi in ambienti altamente distribuiti, dinamici e pervasivi. Dal punto di vista architetturale SAPERE prevede la realizzazione di un substrato spaziale mappato sopra l'infrastruttura di rete pervasiva formato da nodi ciascuno rappresentante una specifica area locale. Ognuno sarà composto principalmente da uno spazio atto alla condivisione delle informazioni con utenti e servizi che in un dato momento si troveranno nella suddetta area. I servizi a disposizione dell'infrastruttura e gli stessi utenti si manifesteranno al sistema inserendo e gestendo negli spazi dei nodi delle rappresentazioni apposite chiamate LSA (acronimo di Live Semantic Annotation), il cui contenuto sarà descritto in termini semantici. La coordinazione tra utenti, servizi e sistema sarà invece garantita da delle leggi di ispirazione chimica dette *eco-laws* che utilizzando gli LSA in veste di reagenti li modificheranno permettendo tramite ciò l'evoluzione di questi e l'emergenza di pattern di coordinazione complessi.

La tesi è volta a espandere e completare il lavoro eseguito nello stesso contesto da due lavori di tesi precendenti [8] [6], con l'obiettivo di ottenere un prototipo che soddisfi appieno le funzionalità previste nello stato attuale del progetto SAPERE. In particolare ciò su cui verte il lavoro qui presentato è lo sviluppo del nuovo modello di coordinazione basato su eco-laws che, rispetto ai lavori precedenti, è l'aspetto piú dinamico e soggetto a modifiche. La fase iniziale ha previsto uno studio della sintassi e semantica del linguaggio delle eco-laws comprensivo di un confronto con le versioni precedenti del linguaggio stesso per determinare i vari aspetti di intervento e la loro fattibilità, seguito poi da una fase di progettazione finalizzata a integrare tali modifiche nel prototipo funzionante realizzato in precedenza. Infine una estesa fase di testing ha concluso il lavoro dimostrando la qualità del nuovo prototipo nel rispetto del modello di coordinazione basato su eco-laws.

La struttura della tesi è organizzata in capitoli come segue. Nel capitolo 1 si trattano brevemente i sistemi pervasivi e si riportano i contributi presenti in letteratura sulla coordinazione ispirata a sistemi naturali. Nel capitolo 2 si presenta SAPERE dal punto di vista architetturale, come descritto in [5], tramite una panoramica degli elementi principali e si descrive il prototipo realizzato nei lavori di tesi precedenti. Il capitolo 3 descrive il linguaggio delle eco-law e il modello operazionale per la loro esecuzione presentato in [11]. Nel capitolo 4 si descrivono i componenti del sistema soggetti a riprogettazione, le loro principali caratteristiche e funzionalità. Il capitolo 5 presenta due casi di studio realizzati sfruttando il prototipo sviluppato. Nel capitolo 6 vengono

riportate le conclusioni maturate sul lavoro svolto e i possibili sviluppi futuri.

Capitolo 1

Pervasive Computing e Sistemi di Ispirazione Naturale

La ricerca nel mondo ICT è negli ultimi anni sempre più indirizzata verso scenari innovativi e altamente complessi. Motore di ciò è la sempre più capillare diffusione di dispositivi portatili dai costi ridotti e dalle alte capacità computazionali quali netbook, smartphone e pda, spesso dotati di sensori in grado di percepire informazioni dall'ambiente che li circonda unitamente alle evoluzioni delle reti che ha permesso un'estesa diffusione della connettività wireless. Prendono il nome di sistemi pervasivi quei sistemi formati da un alto numero di entità (spesso eterogenee) in grado di connettersi e comunicare tra loro e con la capacità di ottenere, tramite semplici e singole mosse elementari delle stesse, il raggiungimento di uno scopo globale.

1.1 Pervasive Computing

Il concetto di pervasive computing è da attribuirsi a Mark Weiser che per primo utilizzo il termine *ubiquitous computing*, suo sinonimo, durante un suo intervento presso Xerox nel 1988 [15]. Secondo la sua innovativa visione il mondo andava incontro alla terza era dei computer: la prima era rappresentata dal periodo dei mainframes, il cui funzionamento era condiviso da numerosi utenti, seguito poi dalla più attuale era dei personal computer dominata dai computer desktop che permettono un rapporto 1 a 1 tra uomo e macchina. L'era del pervasive computing sarebbe invece stata caratterizzata dalla cosiddetta tecnologia calma

that which informs but doesn't demand our focus or attention

ovvero l'integrazione di quest'ultima nelle attività e negli oggetti di tutti i giorni durante i quali noi utilizzatori ci interfacceremo ad un elevato numero di componenti software, più o meno coscientemente. In questa visione tecnologica perciò l'informazione viene percepita e prelevata pervasivamente, interpretata localmente ed utilizzata per gli scopi più vari.

Alla luce di queste potenzialità il passo successivo è indubbiamente la nascita di reti di servizi complesse e pervasive per le quali però sono richiesti metodi di coordinazione e gestione del tutto nuovi. Tali scenari, infatti, sono caratterizzati da caratteristiche quali un altissimo livello di dinamicità e distribuzione e richiedono alle infrastrutture che dovrebbero realizzarli, principalmente, capacità di self-management e self-organisation. Il fatto stesso che gli utenti e i dispositivi in loro possesso fungano contemporaneamente da produttori e consumatori (in gergo *prosumers*, dall'unione delle parole producers e consumers) di ampie moli di informazioni suscita una serie di problematiche da cui è impossibile prescindere e che sono affrontabili solo rivedendo completamente il modo con cui finora i sistemi di servizi sono stati modellati e realizzati.

1.1.1 Sistemi di servizi pervasivi: caratteristiche

Il primo aspetto che caratterizza questa tipologia di sistemi è indubbiamente la *situatedness* ovvero la capacità dei vari componenti di determinare il proprio comportamento fortemente dipendente dalla localizzazione. Il concetto di localizzazione ha un'accezione non solo spaziale ma, spesso e volentieri, anche sociale ovvero in relazione agli altri componenti del sistema. Il concetto di forte integrazione con l'ambiente circostante comporta la necessità che il sistema sia anche reattivo alle modifiche e ai cambiamenti di quest'ultimo, dimostrando caratteristiche di adattabilità, gestione e organizzazione autonome. Questi ultimi requisiti in particolare sono alla base della realizzazione di sistemi con spiccate funzionalità di *fault tolerance*. La robustezza a variazioni di breve termine o malfunzionamenti minori è spesso un requisito sufficiente ma in un sistema ideale sarebbe maggiormente auspicabile anche la capacità di adattamento in seguito a cambiamenti molto consistenti. Tali cambiamenti possono interessare sia la tecnologia che il tipo di servizi offerti. Si dovrebbero tollerare anche evoluzioni di lungo termine dei vari componenti del sistema, dalla struttura interna al tipo di componenti che la formano o ai

pattern di interazione tra servizi e il tutto senza pretendere una riprogettazione del sistema. In ultimo ma non meno importante occorre tenere conto della *apertura* che il sistema deve mostrare nei confronti di servizi e utenti. Ovvero permettere che essi stessi siano in grado di arricchire l'infrastruttura stessa con l'aggiunta di informazioni o anche nuovi servizi, comportandosi come detto precedentemente da prosumers. Questo aspetto diventerà così un altro mezzo di modifica e evoluzione del sistema.

1.2 Ecosistemi di servizi di ispirazione naturale

In natura caratteristiche quali la adattabilità, la robustezza e l'evoluzione di breve e lungo termine sono sempre presenti poiché alla base delle "regole del gioco" che caratterizzano il sistema naturale e regolano le interazioni tra i componenti individuali del sistema. La complessità dei sistemi moderni, anche non ICT, ha fatto sì che l'utilizzo di astrazioni di origine naturale sia sempre più frequente anche se ciò impone un approccio completamente nuovo alla risoluzione dei problemi. Normalmente i sistemi vengono progettati ingegniristicamente come composizione di singole parti più piccole e il cui complesso fornisce risultati predicibili. Al contrario, sistemi di origine naturale e caratterizzati, per esempio, da aspetti di auto-organizzazione mostrano dinamiche che difficilmente possono essere riprodotte in un modello classico. Inoltre si pone il problema di come progettare i singoli componenti in modo tale che il loro comportamento complessivo dia luogo a quello globale voluto. Diversi punti di vista e metodologie sono state teorizzate a seconda del particolare modello naturale che si prende di volta in volta in esame [13]. Quale che sia il modello utilizzato, le caratteristiche chiave che lo definiscono sono raggruppabili in diversi livelli che compongono l'architettura concettuale di un ecosistema di servizi pervasivi di cui una rappresentazione è riportata in figura 1.1.

Il livello inferiore è quello fisico formato da componenti digitali su cui sarà distribuito l'ecosistema, ovvero, una infrastruttura densa di dispositivi informatici collegati tra loro con varie modalità. Al livello superiore, i prosumers accedono al sistema per l'uso e consumo di dati o di servizi. Tra questi due troviamo un livello computazionale astratto che vuole realizzare

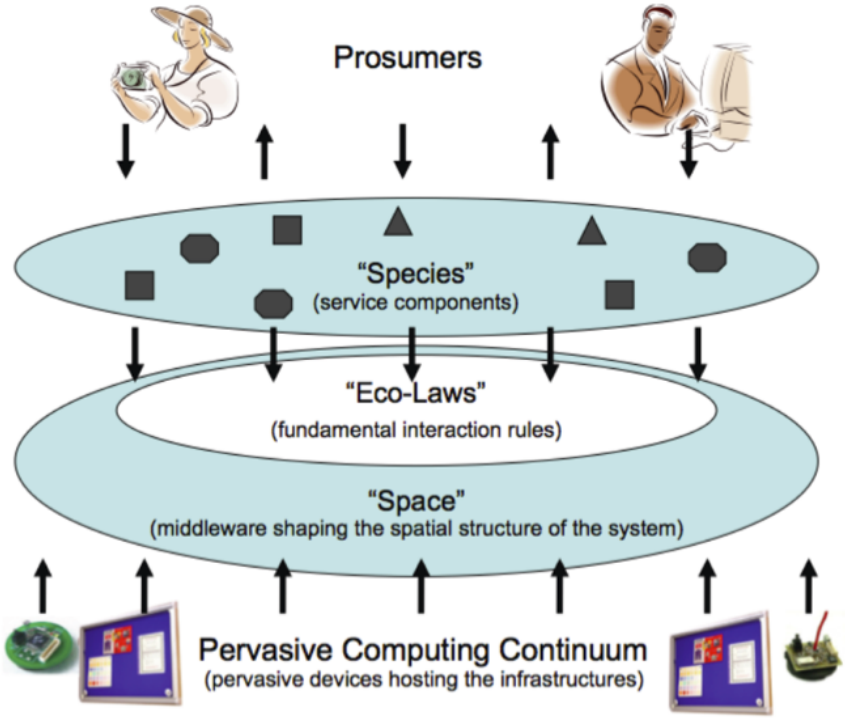

Figura 1.1: Architettura concettuale di un nature-inspired service ecosystem

l'architettura del nostro sistema. Le singole parti che lo compongono hanno il seguente significato generale:

- **Space**
  È il livello che modella l'ambiente in cui gli individui risiedono e li abilita ad interagire e comunicare tra di loro tramite messaggi e attività. Vista la particolare natura distribuita di un sistema pervasivo, ogni individuo risulta situato in una specifica porzione dello spazio e le stesse interazioni che essi intraprendono sono fortemente dipendenti dalla locazione e dalla forma dell'ambiente che li circonda.

- **Eco-Laws**
  É il livello intermedio che definisce le leggi che regolano l'ecosistema nella sua interezza tramite le quali gli individui proliferano e interagi-

	Key Characteristics			Analysis		
	Species	**Eco-Laws**	**Space**	**Adaptive Self-organization**	**Diversity and Evolution**	**Decentralized Control**
Physical	Particles (computational components) and messages (computational fields)	Movements and activities driven by fields (gradient ascent/descent by components)	Network topology or physical space	+ Local and global self-organizing spatial structures can be effectively accommodated	-- Few new components can be accommodated while keeping the laws simple	+ We know well how to build and control specific structures in physics
Chemical	Atoms and Molecules (semantic descriptions represent chemical properties)	Chemical Reactions (matching of semantic descriptions and bonding of components)	Localities (pervasive computing environments)	-- Mostly local self-organizing structures can be effectively accommodated	++ Several new components can be accommodated with the same basic laws	+ Reactants and catalysts can exert control over the dynamics and structure of reactions
Biological	Cells or simple goal-oriented organisms (ants) and pheromones	Diffusion and evaporation of chemical pheromones (determining differentiation of behaviour and activities)	Network topology or physical space	+ Morphogenesis of local shapes, global patterns via movements and pheromones diffusion	+ Reasonable number of new individuals and pheromone flavours can be accommodated without increasing complexity too much	- Mechanisms and control of morphogenesis and biological self-organization not fully understood
Social	Goal-oriented animals (Agents) of various species (Classes) and included passive life-forms (Resources and Data)	Trophic relations (eating), digest, produce, and reproduce	Niches (pervasive computing environments)	+ Local self-organizing structures can be mostly accommodated, although sometimes leading to more global patterns and structures	++ Several new species can be accommodated with the same basic laws	-- Difficult to understand how to enforce control over ecosystems of many species

Figura 1.2: Caratteristiche, vantaggi e svantaggi di approcci nature-inspired

scono. In generale sono semplici leggi che alterano il sistema intervenendo solo su pochi individui e su una porzione minima dello spazio. La composizione di più leggi porta comunque a pattern anche molto complessi.

- **Species**
  Questo livello determina quali sono e come vengono rappresentati gli individui che popolano l'ecosistema. Questi ultimi saranno diversi tra loro ma comunque per permettere interazione dovranno appoggiarsi su una rappresentazione quanto più omogenea.

Di seguito si riassumeranno le caratteristiche generali dei principali approcci evidenziando per ognuno gli aspetti che li caratterizzano in relazione ai tre livelli di cui sopra, conseguentemente si cercherà di determinare quali sono i vantaggi e svantaggi dell'adozione di una particolare soluzione. Quanto definito per ogni sistema è brevemente schematizzato in figura 1.2.

1.2.1 Modelli fisici

Questi tipi di sistemi fondano il loro funzionamento sulla realizzazione degli individui in forma di particelle computazionali immerse in uno spazio fisico avente ben definita metrica per la determinazione delle distanze tra questi ultimi. Il medium di coordinazione è realizzato tramite dei campi computazionali virtuali la cui forma, disposizione, direzione e intensità sono determinanti nel comportamento degli individui. Infatti le leggi naturali sono realizzate in modo tale da modificarli in relazione alla presenza dei suddetti campi. Le metafore a base fisica sono spesso utilizzate in problemi specifici quali il routing spaziale di agenti ma purtroppo non sono adatte ad approcci generali o in cui la diversità richiesta è alta, a meno di non introdurre una maggiore complessità con inevitabile perdita della semplicità nella coordinazione dei componenti.

1.2.2 Modelli chimici

Le metafore chimiche vedono i sistemi sotto forma di raggruppamenti di entità chimiche quali atomi e molecole in spazi localizzati assimilabili a soluzioni in cui possono combinarsi e diffondersi. Ciascuno di questi componenti possiede una descrizione che lo caratterizza tramite la definizione di proprietà che fungono da controparte computazionale alle proprietà chimiche che nella realtà definiscono atomi e molecole. L'interazione e coordinazione dei componenti è mediata da leggi che descrivono reazioni dove in veste di reagenti vengono utilizzati i componenti elementari di cui sopra. Le reazioni definiscono come i componenti vengono trasformati, uniti o collegati per realizzare i pattern e le aggregazioni richieste. In alcuni esempi oltre quanto detto i componenti elementari portano anche una informazione che rappresenta la loro concentrazione nella soluzione, maggiore è la concentrazione di un dato elemento maggiore saranno le sue interazioni con quello che lo circonda. L'aumento o la diminuzione della concentrazione degli elementi può essere sfruttato per realizzare meccanismi di feedback positivi e negativi per la realizzazione di dinamiche complesse quali competizione tra servizi e comportamenti auto-organizzanti. In generale le metafore chimiche sono spesso utilizzate per realizzare la composizione di servizi [10] poiché ben adatte a gestire diversità ed evoluzione dei sistemi, potendo addirittura influenzarne il comportamento con l'utilizzo di entità pensate come catalizzatori. Il rovescio della medaglia è che risulta complicato gestire gli aspetti di distribuzione

spaziale che caratterizzano i sistemi pervasivi poiché in generale le regole chimiche esprimono modifiche a livello di singolo composto e quindi localmente o al più nel proprio vicinato.

1.2.3 Modelli biologici

Gli approcci a base biologica prendono spunto da sistemi formati da piccoli organismi o colonie di insetti. Gli individui in queste astrazioni sono caratterizzati da ridotte capacità e sprovvisti di particolare intelligenza e sono guidati da comportamenti reattivi ai segnali che ricevono dall'ambiente in cui sono situati. Tali stimoli sono solitamente di natura chimica, per esempio nella stigmergia viene rappresentato dalla presenza di feromoni che sono rilasciati involontariamente da ogni individuo nello spazio che li circonda. Tali approcci permettono anche in questo caso la realizzazione di pattern globali a partire da minime interazioni locali facilmente applicabili in ambito distribuito. Sistemi realizzati in questo modo soffrono, però, della possibilità di poter definire solo un ridotto numero di possibili tipologie di interazione limitando le meccaniche di coordinazione realizzabili e mancando delle caratteristiche di evoluzione e adattività di lungo termine richieste al sistema.

1.2.4 Modelli sociali

La categoria dei modelli sociali si riferiscono in parte alle metafore biologiche di cui sopra, ma osservandole da un livello più alto con l'introduzione di una differenziazione degli individui presenti nell'ecosistema. Le interazione che vengono principalmente realizzate in questi modelli sono quelle di tipo trofico ovvero concernenti la posizione degli individui all'interno della catena alimentare. Pertanto ogni entità del sistema rappresentato come appartenente a una specie ben precisa e con un fine definito, baserà il suo comportamento sul recupero di cibo che sarà l'astrazione di una certa tipologia di risorse. Tali risorse di cui si necessita per sopravvivere o prosperare possono essere rappresentate da specifici servizi o dati. Le leggi naturali che descrivono il regolamento dell'ecosistema prevederanno come è organizzata la catena alimentare, descrivendo requisiti e condizioni a cui sottostare affinché gli individui possano recuperare cibo, nutrirsene, riprodursi e interagire con i rappresentanti delle altre specie, andando a generare modifiche locali nel sistema. Normalmente l'ecosistema è visto suddiviso in compartimenti as-

similabili a biosfere in comunicazione tra loro, così da permettere anche il passaggio di entità tra una e l'altra. Vantaggi di una simile astrazione sono prevalentemente la capacità di poter realizzare egregiamente sistemi caratterizzati da funzionalità di auto-organizzazione spaziale e di evoluzione di breve e lungo termine tramite la competizione tra individui. La controparte negativa risiede nella difficoltà di gestire l'equilibrio locale e globale di sistemi reali di questo tipo che risulterebbe in un aumento ingente della complessità computazionale [13].

1.2.5 Approcci ibridi

Per quanto appena visto, ogni singolo modello presenta vantaggi e svantaggi in vari aspetti del suo utilizzo rivelandosi in certi aspetti ottimo ma allo stesso tempo povero o incompatibile ad altri utilizzi e rappresentazioni. La scelta quindi di un modello per gestire sistemi di servizi pervasivi non è assolutamente compito semplice e spesso ci si rivolge verso scelte dalla natura ibrida che attingono caratteristiche da più di un modello cercando di comporre i benefici delle singole. Il lavoro di questa tesi si basa sul modello ibrido delineato in [9, 13, 14] in cui la metafora usata è di tipo bio-chimico dove la diversità, l'evoluzione e il controllo sono gestiti con approccio ispirato ai modelli chimici, mentre la gestione della località e distribuzioni spaziale del sistema è gestito con un modello biologico come esplicato al paragrafo seguente.

1.3 Coordinazione con spazi di tuple biochimici

Una proposta di modello per un framework per la coordinazione di un ecosistema di servizi pervasivi ad approccio ibrido è teorizzato in [9], dove si introduce il concetto di spazio di tuple biochimico in cui alle normali tuple viene associato un valore indice della grado di pertinenza (o attività), che viene correlato al concetto di concentrazione chimica ed è determinante nei meccanismi di coordinazione. Infatti, ad esempio, una tupla con concentrazione relativamente bassa sarà pressoché inerte e utilizzata molto meno frequentemente dai meccanismi di coordinazione rispetto una con una concentrazione maggiore. Lo spazio conterrà inoltre delle leggi di tipo chimi-

co che andranno ad evolvere la concentrazione delle tuple nel tempo, nello stesso modo in cui avviene per sostanze chimiche presenti in una soluzione con l'emergenza di pattern chimici, anche complessi, con proprietà di auto-organizzazione. L'estensione del modello verso l'aspetto biologico fa si che tali leggi siano potenziate con la possibilità di definire meccanismi per lo spostamento, o meglio diffusione, delle sostanze chimiche attraverso i vari compartimenti biologici che formano il sistema.

La scelta di utilizzare il paradigma degli spazi di tuple, ormai studiata a fondo e ben collaudata, permette di realizzare spazi condivisi in cui numerosi agenti possono accedere autonomamente per realizzare interazioni che si basano sulla scrittura, lettura e modifica di informazioni. Il modello di partenza è un estensione stocastica del framework *Linda* in cui si aggiunge alle tuple l'informazione della loro concentrazione, rifacendosi a sua volta a un modello antecedente di tipo probabilistico [3] in cui le tuple erano estese con l'aggiunta di "pesi" che influivano sulla probabilità del loro recupero. Nello stesso modo anche le primitive dovranno essere modificate per permettere di specificare l'informazione relativa alla concentrazione delle varie tuple, per esempio tramite una *out* si dovrà poter indicare oltre la tupla stessa da inserire anche la sua concentrazione iniziale e nel caso una esatta copia della tupla sia già presente nello spazio le relative concentrazione verranno sommate. Stesso dicasi per la primitiva duale *in* con la quale indicando una concentrazione si impone di recuperare dallo spazio una data tupla in quantità indicata andando conseguentemente a diminuire la concentrazione nello spazio della stessa. Per quanto riguarda la parte distribuita, il sistema è pensato come una rete di compartimenti biologici in cui il passaggio delle entità da uno all'altro è permesso poiché specificabile come prodotto di una eco-law.

1.3.1 Catene Markoviane Tempo-Continue

Il modello è basato sul framework delle CTMC (*Continuous-Time Markov Chains*) che caratterizzano sistemi con transizioni periodiche tra uno stato e un altro per processi casuali in cui le transizioni possibili, in un dato momento, sono solo dipendenti dello stato attuale e non da quelli passati. Una CTMC viene rappresentata come un grafo in cui i nodi sono gli stati del sistema in esame e gli archi che li congiungono sono transizioni del sistema annotate con dei valori reali positivi definiti *rate*. Tale valore rappresenta la frequenza media della transizione assumendo che la distanza temporale tra

due occorrenze della transizione segua una distribuzione di probabilità esponenziale negativa. La scelta di utilizzare questo particolare meta-modello stocastico deriva dai lavori di Gillespie [4] che teorizzò la possibilità di simulare con successo il comportamento di reazioni chimiche tramite CTMC. Ogni reazione chimica sarà la transizione di un particolare processo markoviano da un particolare stato di partenza ad uno finale nel quale si avrà la diminuzione della concentrazione di alcuni composti e l'aumento di altri. Conseguentemente ogni reazione dovrà possedere un rate che ne definisce la frequenza media di attivazione, anche se la frequenza effettiva di attivazione è dipendente anche dalla concentrazione di ogni reagente nella soluzione. Ad ogni passaggio della computazione si dovranno determinare tutti i rate delle reazioni attuabili sulla base dei reagenti presenti nella soluzione e scegliere probabilisticamente quello da attuare. Tale probabilità sarà maggiore per le reazioni che hanno rate più alti, si determina l'istante di tempo Δt in cui essa avverrà tramite la formula $log(\frac{1}{\tau})/R$, con τ valore casuale reale tra 0 e 1, a partire dall'istante corrente.

1.3.2 Eco-law

Come già detto in precedenza le leggi che devono caratterizzare il comportamento del suddetto sistema traggono ispirazione dal modello chimico. Si tratta di leggi che permettono di indicare i reagenti interessati, i prodotti generati a partire dalla reazione e il rate con cui la stessa viene attivata. La forma tipica con cui si descrivono è qualcosa del tipo $\texttt{A} + \texttt{B} \xrightarrow{r} \texttt{C}$ in cui si intende che una molecola di tipo `A` che si incontri con una di tipo `B` (mediamente con frequenza r) possano interagire per realizzare un elemento di tipo `C`. In realtà il rate effettivo con il quale tale reazione si attiverà è funzione non solo di r ma anche delle concentrazioni di `A` e `B` poiché deve prendere in conto il numero di possibili combinazioni delle molecole in gioco e, per esempio, siano n_a e n_b le concentrazioni dei suddetti allora l'effettivo rate della reazione è calcolabile come $r * n_a * n_b$ confermando che quanto più è alta la concentrazione degli elementi tanto più probabile sarà l'attivazione della reazione. Inoltre l'effetto comporterà la diminuzione di una unità delle concentrazioni n_a e n_b e aumenterà contestualmente quella di n_c (riferito alla molecola `C`). Tale modello è molto potente e permette di rappresentare qualsiasi tipo di reazione chimica ma non solo, per esempio una legge del tipo $\texttt{X} + \texttt{Y} \xrightarrow{r} \texttt{X} + \texttt{X}$ può essere usata per modellare una dinamica seconda

la quale il predatore `X` si nutre di una preda `Y` e conseguentemente genera un figlio. Tali leggi dovranno pertanto esprimere, in un linguaggio specifico, i templates (o patterns) di ogni reagente e occorrerà prevedere un engine capace di determinare la somiglianza tra essi e le tuple presenti nello spazio. A tale scopo si suppone l'esistenza di una funzione μ di matching che restituisce un valore compreso tra 0 e 1 che indica la somiglianza tra un tupla e un template, maggiore è la somiglianza maggiore il risultato di tale funzione. L'obiettivo è di utilizzare tale informazione per regolare l'esecuzione delle eco-laws stesse interpretando la somiglianza come un punteggio di bontà e modificare di conseguenza il rate delle reazioni favorendo quelle per cui il match tra reagenti e tuple è maggiore.

1.3.3 Rappresentazione dei dati e match semantico

Punto centrale del meccanismo delle reazioni è rappresentato dalla funzione di match μ tra reagenti e template definita al punto precedente. Tale funzione viene prevista in [9] come application specific pertanto dovrà di volta in volta essere rimappata a seconda della effettiva funzionalità che si desidera sperimentare. La versione più semplice sarà quella di mero match sintattico ove il punteggio potrà assumere solo due valori distinti: 0 nel caso template e tupla differiscano o 1 nel caso i due siano esattamente uguali. Per soddisfare però i requisiti di apertura richiesti al sistema ciò non risulta sufficiente. È infatti impensabile poter prevedere con anticipo tutti i tipi di tuple che il sistema dovrà gestire nelle eco-law, volendo organizzare una coordinazione per un ampissima gamma di servizi, dati e individui quanto più eterogenei. L'ottimale sarebbe raggiungibile potendo suddividere le tuple del sistema non secondo la loro rappresentazione sintattica ma bensì secondo il concetto ontologico che sottintendono. Alla luce di ciò occorre introdurre il concetto di match semantico tramite l'utilizzo di particolari ontologie da utilizzarsi in funzione del dominio applicativo del sistema. Con questa meccanica e l'utilizzo di reasoner in grado di gestirli sarà possibile correlare informazioni sintatticamente diverse ma semanticamente simili e poter definire opportunamente un operatore μ come quanto richiesto precedentemente. Ciò fa si che nonostante le tuple potranno assumere contenuti sintatticamente dissimili le leggi potranno comunque essere applicate perché descritte in termine di concetti semantici.

Il lavoro di questa tesi parte da un prototipo iniziale sviluppato in [8], e raffinato in [6] seguendo i miglioramenti proposti in [5]. Il modello in questione ha subito nuove modifiche e evoluzioni presentate successivamente in [12], [7] e [11] che la tesi in esame si è posta in obiettivo di realizzare. Il modello dell'infrastruttura ha esso stesso, pertanto, subito nel corso degli ultimi mesi varie modifiche che lo hanno in parte discostato per alcuni aspetti da quanto esplicato in questi paragrafi introduttivi che restano però il punto di partenza e il background teorico di quanto sviluppato.

Capitolo 2

SAPERE Project

Il progetto SAPERE, che sta per Self-Aware Pervasive Services Ecosystems [2], ha come obiettivo lo sviluppo di un framework per la coordinazione decentralizzata di servizi da realizzarsi in scenari di pervasive computing e sistemi fortemente distribuiti. Il progetto, avvalendosi degli sviluppi degli ultimi anni nella ricerca basata su sistemi di ispirazione naturale, intende promuovere un innovativo metodo di concepimento e progettazione delle architetture service-oriented e degli algoritmi associati. Il principale paradigma usato è quello degli ecosistemi di servizi con modello di coordinazione pensato come ibrido del modello chimico e biologico e con l'obiettivo di realizzare i comportamenti autonomici delle varie entità presenti non come feature degli stessi ma come caratteristiche inerenti e definite all'interno dell'ecosistema preso nella sua interezza. Ciò sarà realizzabile tramite un modello delle interazioni che promuove principalmente la context-awareness delle entità e l'introduzione di algoritmi decentralizzati di self-management e self-organisation.

2.1 Caso di studio

Per motivare e meglio inquadrare gli obiettivi posti dal progetto si esemplifica un caso di studio di uno scenario specifico che ci si aspetta il sistema da svilupparsi possa gestire. Si immagini uno scenario in cui l'ecosistema sia popolato da display, utenti forniti di PDA e/o smartphone e servizi pubblicitari o informativi. Uno scenario di questo tipo è quanto mai attuale vista la repentina evoluzione che negli ultimi anni ha avuto la tecnologia mobile e delle connessioni wireless unitamente all'effettiva capillare diffusione di schermi

all'interno, in particolare, dei luoghi pubblici più frequentati delle aree urbane quali centri commerciali, stazioni, aeroporti e uffici. Normalmente tali display visualizzano informazioni preconfigurate o gestite manualmente sulla base del loro posizionamento, ad esempio mostrando slides cicliche di pubblicità all'interno di un centro commerciale o l'elenco delle partenze e arrivi della giornata nella stazione ferroviaria. Ciò che è certo è che tali informazioni sono indipendenti dalla presenza degli utenti nell'area.

Il migliore sfruttamento, invece, che si potrebbe fare di un tale dispiego di mezzi sia nei confronti degli utenti che dei provider delle informazioni e pubblicità sarebbe quello di realizzare un approccio nuovo e legato al contesto in cui tali display sono immersi. Ovvero i display dovrebbero adattare il contenuto da loro mostrato sulla base degli utenti che possono potenzialmente osservarle e fruirne e sulle necessità che tali utenti potrebbero mostrare in un dato momento. Nell'ipotesi pertanto di trovarsi in una stazione ferroviaria o in un aeroporto, le informazioni da visualizzare potrebbero essere legate alla loro prenotazione indicando per esempio informazioni quali il binario di partenza del treno o il gate per l'aereo o eventuali variazioni di orario che si potrebbero essere verificati. Perché però ciò sia possibile occorre prevedere un modo tramite il quale gli utenti possano fornire informazioni al sistema che poi efficacemente questi possa usare per adattare e arricchire i messaggi mostrati. Ancora meglio tale meccanismo dovrebbe essere quanto più trasparente all'esperienza dell'utente e possibilmente del tutto indipendente dalla propria volontà. Oltre a ciò sarebbe interessante che il sistema oltre a gestire opportunamente tali informazioni sia in grado di rilocarle e modificarle nel caso alcune condizioni vengano a mancare o si alterino come per esempio con l'aggiunta di nuovi display o il verificarsi di malfunzionamenti su quelli già presenti. Anche l'evoluzione in termini tecnologici dei display o dei sensori che fanno parte del sistema deve essere tollerato e gestibile, permettendo così che si possano utilizzare nuovi dispositivi dalle feature migliori.

Utilizzando il modello teorizzato al capitolo precedente (cfr. 1.3) basato sull'approccio biochimico, non solo le informazioni, ma anche i servizi, gli utenti e i dati dell'ambiente saranno modellati sotto forma di sostanze chimiche dotate di determinate caratteristiche: per i display potranno essere la posizione e l'orientamento, per gli utenti le preferenze personali e i dati anagrafici e così via. Tramite regole di astrazione chimica si potrà descrivere le interazioni tra le entità che nel caso in studio vogliamo siano la lettura delle preferenza di utente entro un certo raggio dalla posizione del display;

in seguito a ciò un servizio informativo corrispondete alle informazioni lette sarà scelto per essere *associato* al display che, consapevole di ciò, ne preleverà il contenuto e inizierà a mostrarlo. Questa è una delle più semplici coordinazione che un sistema del genere dovrebbe svolgere ma i potenziali casi di studio sono molto numerosi. Per esempio si potrebbe pensare di modificare il caso esposto prevedendo che il contenuto della visualizzazione sia un video o una presentazione non statici e che si disponga di una serie affiancata di display lungo un corridoio. Il modello di coordinazione deve fare sì che al passaggio di un utente il servizio sia visualizzato sul display più vicino a lui e lo segua nel suo movimento lungo il corridoio in modo tale che però la visione risulti continua e fluida ovvero non venendo reiniziata ad ogni cambiamento di schermo.

2.2 Modello architetturale

Dal punto di vista architetturale SAPERE è identificato da un substrato spaziale mappato immediatamente sopra la struttura di rete che può essere composta anche da un insieme molto denso di dispositivi hardware. Questo livello è chiaramente caratterizzato da fortissima eterogeneità e dinamicità poiché nuovi device possono essere aggiunti così come alcuni possono essere rimossi o spostati dinamicamente modificando in parte la topologia.

Ogni componente presente nel mondo reale ha una relativa rappresentazione semantica all'interno dell'ecosistema chiamata LSA (acronimo di Live Semantic Annotation, si veda 2.2.2). Tali rappresentazione riflettono costantemente lo stato e il contesto del componente che rappresentano e solo tramite la loro osservazione e modifica è possibile interagire con gli individui che rappresentano. Gli LSA non vengono utilizzati per modellare unicamente le entità concrete presenti nell'ambiente ma anche quelle astratte quali le informazioni provenienti dagli utenti, eventi, dati rilevanti per il sistema, i servizi e altre tipologie di componenti software. In questo modo ognuno di questi può essere visto come un *singolo individuo* del sistema con una rappresentazione uniforme. Essendo gli LSA associati a determinate entità spazialmente localizzate il substrato è suddiviso in vari nodi computazionali in cui tali LSA risiedono e vi vengono automaticamente iniettate alla manifestazione di un nuovo componente così come l'infrastruttura ha il compito di gestirne la rimozione quando il componente abbandona il nodo. Inoltre i

vari nodi sono collegati tra loro sulla base di un concetto di prossimità fisica o logica e mantengono traccia di tutti i nodi a loro "vicini".

All'intermo dei nodi, la dinamica del sistema è invece guidata dalle *eco-laws* (cfr. 2.2.3 e 3), una sorta di reazioni chimiche virtuali tra gli LSA dei vari individui. La scelta di questi ultimi è determinata da meccanismi di pattern-matching ovvero le eco-laws definiscono template per ogni "reagente" e solo gli LSA che rispecchiano tali template possono essere scelti in loro veste. Le eco-laws, quindi, sono il vero motore che porta alla aggregazione, coordinazione, interazione e nascita di legami tra gli individui definendo il modo con cui questi si modificano ed evolvono e possono comportare anche la creazione di nuovi LSA o lo spostamento da un nodo all'altro. In definitiva il sistema promuove la filosofia: "ogni componente è un LSA, ogni interazione è attraverso le eco-law".

Da tale aspetto risulta che la coordinazione dei servizi e il comportamento adattivo all'interno del sistema non sono responsabilità del singolo individuo, ma invece emergono dalla dinamica del sistema nella sua interezza. Infatti l'unione delle eco-laws e dei legami chimici virtuali da esse realizzati con i meccanismi interni goal-oriented di agenti, servizi e utilizzatori del sistema farà sì che le dinamiche del sistema non siano al livello del singolo componente ma piuttosto promuovano una sorta di "self-awareness sistemico". La garanzia di adattabilità così definita può conseguentemente permettere anche la tolleranza di evoluzioni sistemiche di lungo termine. Infatti anche se non si fanno assunzioni sulla capacità evolutiva dei singoli componenti, sarà l'iniezione di nuovi componenti aggiornati nel sistema, automaticamente incorporati nella dinamica di coordinazione, che realizzerà di fatto una sorta di evoluzione. Infatti, come per la selezione naturale, sono le specie di componenti ad evolvere e non i singoli individui: classi di componenti più adatti nello svolgimento di certi compiti rimpiazzeranno quelli più obsoleti come conseguenza delle dinamiche dell'ecosistema, senza particolari interventi all'infrastruttura. L'immagine 2.1 rappresenta l'architettura logica di riferimento del sistema nel suo complesso.

Dal punto di vista implementativo il framework SAPERE vuole essere un middleware minimale, che reifica il concetto di LSA tramite l'utilizzo di tuple semantiche che saranno dinamicamente memorizzate e aggiornate in spazi appositi disposti all'interno di nodi spazialmente situati su alcuni device di cui il network è composto. Le eco-law e il motore che le regola e applica sarà eseguito su ciascuno dei singoli nodi per promuovere in maniera

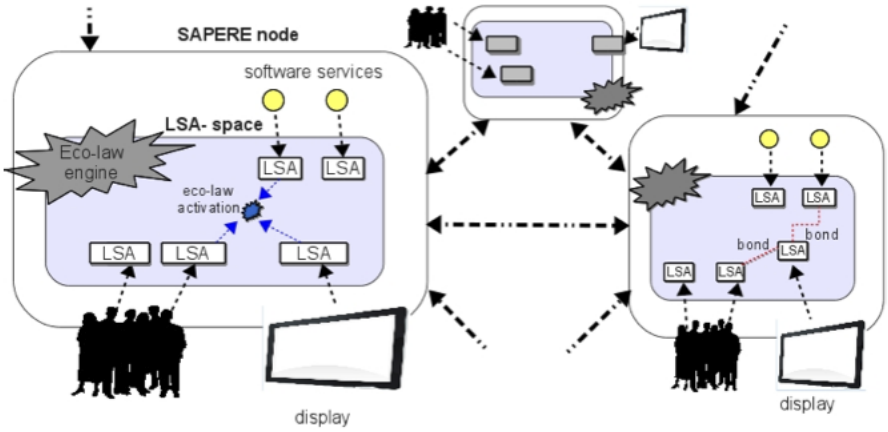

Figura 2.1: Architettura logica di SAPERE

trasparente i meccanismi di coordinazione. Dal punto di vista del singolo nodo, la struttura ricorda quello di uno spazio di tuple ma la memorizzazione e l'applicazione delle eco-law così come anche alcune delle primitive di accesso a esso si differenziano non poco da quest'ultimo. Nelle sezioni seguenti si scende maggiormente nei particolari dei singoli aspetti precedentemente citati.

2.2.1 Node

Ogni nodo non è una entità monolitica ma è composto da un insieme di moduli ognuno pensato per un compito specifico quali possono essere la gestione degli LSA o l'applicazione delle eco-law.

Innanzitutto il nodo fornisce una **interfaccia** che mette in grado agli individui esterno ad esso (utenti, servizi o altri nodi) di interagire con esso. Le operazioni offerte sono `inject` per inserire un nuovo LSA al suo interno, `update` per modificarne uno, `remove` per rimuoverlo e la `read` e `observe` per leggerlo e osservarlo. Da notare, come detto in precedenza, che per quanto simile a uno spazio di tuple le primitive di accesso sono diverse in quanto, per esempio, mentre in uno spazio normale le operazioni sfruttano ampiamente il pattern-matching per il recupero o la lettura di una tupla che segue il pattern specificato, le operazioni permesse sono basate sulla conoscenza

dell'identificatore del LSA interessato univoco all'interno del sistema (cfr. 2.2.2). L'interazione con altri nodi dell'ecosistema è garantita per permettere il riconoscimento del proprio vicinato poiché alcune eco-laws prevedono lo spostamento di LSA nei nodi adiacenti a quello in cui vengono attivate.

Uno dei componenti centrali è rappresentato dal **LSA-Space** ovvero lo il componente passivo, simile a uno spazio di tuple, pensato per la memorizzazione degli LSA locali. Esso fornisce un interfaccia che permette la *aggiunta*, *rimozione*, *modifica* e *lettura* degli LSA contenuti al suo interno gestendo aspetti tecnici legati a ciò quali l'assegnamento degli identificatori univoci agli LSA e indicizzazione di essi per un più veloce recupero. L'accesso a tale componente è comunque un possibile punto di failure se non opportunamente gestito poiché soggetto a problemi di accesso concorrente ai dati.

Il cuore vero e proprio del nodo è però rappresentato dal **Reaction Manager**, componente che mantiene al suo interno le eco-law applicabili nel nodo (che non sono necessariamente le stesse presenti in altri nodi dello stesso ecosistema) e si occupa della determinazione di quando una di esse è attivabile procedendo ad operare le modifiche opportune sul LSA-space. Per fare ciò gestisce internamente a sé stesso una lista di eventi per ognuno dei quali corrisponde un operazione che può essere di tipo esterno o interno. Le prime sono richieste di operazioni su un LSA (iniezione, rimozione, modifica o osservazione) e sono prioritarie mentre le seconde sono attivazioni di specifiche eco-law che attendono di essere eseguite. L'esecuzione di una eco-law comporta la modifica di parte degli LSA dello spazio che sono stati scelti come reagenti per tale attivazione. Volendo simulare un comportamento chimico si utilizza il modello delle CTMC (si veda 1) e perciò ogni eco-law possiede un tempo di attivazione specifico calcolato a partire dal *rate* che tale reazione deve avere. Tutte le reazioni attuabili sono quindi ordinate per tempo di occorrenza.

La scelta di quali eco-law attivare e quando farlo è demandata dal Manager al componente che prende il nome di **Reagent Matcher**. Questo si occupa, sotto richiesta del Manager di processare le eco-law definite al suo interno e trovare tutte quelle attivabili sull'insieme di LSA presenti nello spazio interno. Ciò è possibile tramite operazioni di filtraggio degli LSA sulla base dei reagenti definiti nelle eco-law e su varie fasi di pattern-matching, anche complesse. Per ogni reazione riconosciuta come attivabile il Matcher fornisce anche le operazioni da eseguire sugli LSA dello spazio per realizzarla.

2.2.2 LSA

Gli LSA, Live Semantic Annotation, sono entità utili a rappresentare tutto i componenti del sistema. Sono l'entità elementare su cui tutto il framework di *SAPERE* appoggia poiché possono sia rappresentare device e quindi un utente presente nel sistema ma anche ad un servizio offerto dall'infrastruttura oppure una informazione che uno di questi due può aver immesso nel sistema stesso. In concreto ciò che realizza è un handle uniforme per qualsiasi elemento che può essere presente nel sistema, indifferentemente dalle sue caratteristiche interne e ne fornisce una interfaccia osservabile. Sempre tramite gli LSA si ha l'interazione con gli altri componenti tramite principalmente operazioni di update.

Sono definiti Live proprio per la loro capacità di rappresentare la dinamicità associata allo scenario e ai singoli componenti cui sono riferiti, che nella maggior parte dei casi sappiamo essere entità mutevoli e il cui stato o contesto può cambiare anche molto frequentemente. Per esempio basti pensare che ogni LSA di un device possegga informazioni sulla sua posizione geografica già ci si rende conto come necessitino di essere frequentemente aggiornate da cui il concetto di "vita" legato ad esse. Gli LSA hanno anche una accezione semantica: i dati che rappresentano devono essere possibilmente interpretati seguendo una particolare ontologia. Questo fa sì che non sia necessario un accordo a priori sulla sintassi utilizzata ed una conoscenza dei tipi di entità rappresentabili, purché esista una ontologia che descriva appropriatamente la conoscenza e i concetti a essi relativi. Tale ontologia infatti, se ben sfruttata di compiere inferenze sui dati rappresentati e di cogliere relazioni reciproche interpretandoli secondo il giusto significato a prescindere dal significante utilizzato.

Il contenuto degli LSA viene definito in termini di associazioni tra proprietà e valori. Non si fanno assunzioni sul tipo di valori memorizzati negli LSA, ma si può assumere che possano essere rappresentati tramite delle stringhe senza perdere di generalità. Volendo si potrebbe pensare di utilizzare tecnologie quali RDF (Resource Definition Framework, cfr 3) per indicare le proprietà e i valori da essi contenuti così che possa essere definito contestualmente al valore anche il tipo. Ogni proprietà, come per RDF, è multivalore e ogni proprietà non definita in un LSA si ritiene essere presente con un set vuoto di valori assegnatogli.

L'uniformità che la particolare rappresentazione degli LSA viene a for-

mare su tutti gli elementi del sistema pervasivo fa sì che ciascuna di esse possa essere scelta come reagente delle eco-laws, sotto certe condizioni come spiegato alla sezione successiva.

2.2.3 Eco-law

Mentre i vari componenti attuano il loro comportamento individuale osservando il loro contesto e modificando opportunamente i loro LSA, il *comportamento globale* del sistema è guidato da un sistema di leggi simil-chimiche chiamate eco-law. Esse possono essere viste come regole avanzate di riscrittura che si basano su template (o pattern) relativi agli LSA presenti nello spazio, l'esecuzione di una di queste comporta degli effetti sul LSA-Space in termini di aggiunta, modifica, rimozione o spostamento di LSA in esso. Alla luce di ciò anche le stesse operazioni primitive sullo spazio possono essere viste come particolari eco-laws caratterizzate da un rateo istantaneo (quindi esecuzione immediata) e dotate di soli prodotti, ovvero l'operazione stessa.

Seguendo la similitudine chimica che rappresenta reazioni con la notazione $\mathtt{A} + \mathtt{B} \longrightarrow \mathtt{C} + \mathtt{D}$, secondo la quale gli elementi indicati sulla sinistra vengono *sostituiti* da quelli a destra, un eco-law è rappresentata da una relazione nella cui parte sinistra vengono indicati i reagenti e nella destra i prodotti. I reagenti saranno i pattern con cui selezionare appropriati LSA sullo spazio mentre i prodotti rappresentano, con una sintassi equivalente ai reagenti, gli effetti causati dall'esecuzione di tale reazione virtuale. Un esempio di una possibile eco-law è:

```
?U:[type=(user), context=?C] +
?D:[type=(display), context has-not ?C, contextualizing=(yes)]
-mark(0.5)-> ?U + ?D:[context has ?C]
```

I letterali preceduti da ? indicano delle variabili. Ogni reagente è indicato da una variabile iniziale che accoglierà l'identificatore del LSA scelto in sua vece ed è poi seguito da un'espressione racchiusa tra `[ ]` che definisce dei filtri sulle proprietà che l' LSA deve possedere. Ogni filtro ha una forma del tipo *p op v* dove *p* definisce una proprietà, *v* un valore e *op* un tipo di filtro: *p* e *v* possono assumere valori o diversi tipi di variabili mentre il filtro determina la condizione richiesta su essi. Nella parte destra sono invece riportati i prodotti, scritti con la stessa sintassi dei reagenti ma interpretati come operazioni da eseguire (spesso sui reagenti stessi) come effetto della eco-law. Si noti

anche la presenza nel separatore tra reagenti e prodotti l'indicazione che la reazione è di tipo *markoviano* con rateo 0.5. Volendo quindi dare un significato all'esempio, l'eco-law riportata seleziona due LSA, il primo di tipo `user` con una proprietà `context` assegnata a un certo valore `C` e un secondo LSA di tipo `display`, a sua volta in possesso di una proprietà `context` ma priva del valore `C` e una `contextualizing` posta a `yes`. Il prodotto atteso di tale reazione sarà quello di aggiungere il valore `C` alla proprietà `context` del secondo LSA lasciando il primo completamente immutato. Le condizioni poste su un reagente da una eco-law sono necessarie e sufficienti, con ciò si intende che un LSA per il quale tali condizioni sono rispettate è utilizzabile in sua vece e la presenza di altre proprietà non menzionate tra i filtri non inibisce tale fatto.

La sintassi e semantica completa delle eco-laws, evolutasi rispetto alla definizione iniziale di [5] e descritta per esteso in [11], è molto più complessa di quanto esemplificato e viene discussa al capitolo 3 poiché è sulla sua implementazione nel progetto che verte maggiormente il lavoro realizzato in questa tesi.

2.3 Analisi del prototipo iniziale

Come già detto in precedenza, il lavoro svolto nel corso della tesi prevede lo sviluppo del framework per la creazione e esecuzione delle eco-law e per fare ciò si appoggia e riutilizza parte del lavoro svolto in tesi precedenti (in particolare [6]) così da partire da un prototipo già in parte funzionante. In questa sezione si vuole presentare brevemente il prototipo iniziale evidenziando alcune delle scelte implementative realizzate e indicando i principali aspetti su cui si dovrà intervenire durante lo sviluppo dei nuovi componenti.

2.3.1 Architettura del nodo SAPERE

L'architettura del nodo sviluppato nel prototipo è riportata in figura 2.2. Come si nota la sua struttura segue da vicino quanto descritto alla sezione precedente mostrando elementi del tutto equivalenti.

Lo Space è un componente passivo realizzato in maniera similare a uno spazio di tuple con la funzione di repository per gli LSA del nodo. Mette a disposizione metodi per recuperare, modificare, inserire e rimuovere LSA da esso.

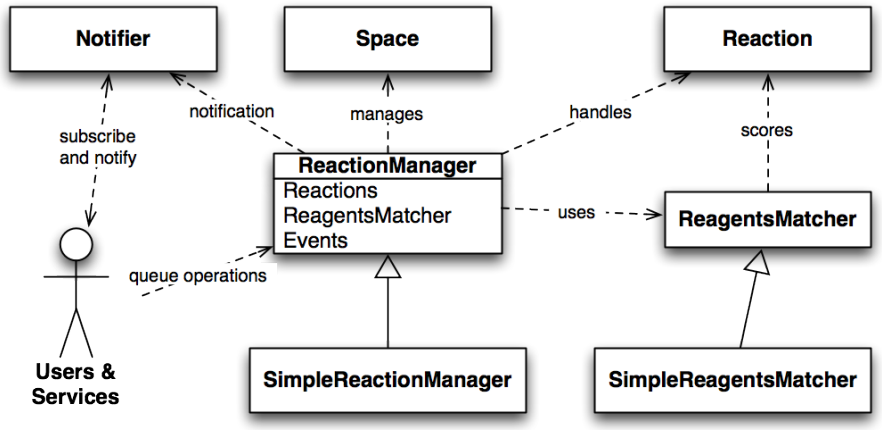

Figura 2.2: Architettura interna di un nodo del prototipo e interazioni

Il componente di maggiore interesse è il ReactionManager. A discapito di quanto potrebbe suggerire il nome questa entità svolge diversi compiti, tutti di una certa importanza nella dinamica di funzionamento del nodo. Innanzitutto è l'unico componente attivo in grado di eseguire operazioni sul LSA-Space: tale scelta è stata dettata principalmente per evitare problemi di accesso condiviso alle risorse rappresentate dagli LSA. Questo ha permesso di limitare la complessità di buona parte dei componenti del sistema ma fa sì che il Manager diventi anche un collo di bottiglia poiché le operazioni da eseguire sullo spazio che rappresentano l'interazione degli utenti dell'ecosistema saranno di fatto sequenzializzate. In aggiunta a ciò il Reaction Manager gestisce la logica che gli di dà il nome ovvero quella dell'attivazione e esecuzione delle eco-law sullo spazio. Dovendo simulare un sistema CTMC ciò viene fatto modellando lo scorrere del tempo come una successione di eventi da verificarsi in un dato istante temporale denominati *SapereEvent*. Ciascuno di essi rappresenta un operazione esterna richiesta sullo spazio o un trasformazione interna degli LSA in seguito all'attivazione di una eco-law. Quale che sia il loro significato il Manager mantiene e gestisce una lista di eventi ordinati per tempo di occorrenza che consuma progressivamente.

Ogni evento eseguito modifica parte dello spazio e conseguentemente può portare alla rimozione di eventi non più attuabili o alla creazione di nuovi. Il compito di determinare quali reazioni sono attivabili e quali non lo sono più

(per modifica o rimozione di alcuni LSA) è demandato al ReagentMatcher, un componente reattivo che su richiesta del Manager determina il matching tra gli LSA presenti nello spazio e le eco-law del nodo. Questa operazione è molto complessa e prevede fasi successive di confronto tra pattern e LSA e di assegnamenti e sostituzioni di variabili. Per ogni reazione completamente eseguibile il Matcher restituisce l'evento relativo comprensivo dell'istante di tempo di attivazione e delle operazioni che occorre eseguire sullo spazio perché si realizzi. Tale componente è probabilmente il più complesso, dovendo eseguire il reasoning legato alla scelta degli LSA come reagenti delle reazioni ed è direttamente dipendente dalla complessità ed espressività del linguaggio delle eco-law.

Tra le possibili operazioni eseguibili sullo spazio da un agente esterno oltre quelle di inserimento, rimozione e modifica degli LSA vi è anche quella di *observing* per poter essere comunicati di ogni cambiamento subito da un determinato LSA. Questa funzione è gestita da un componente esterno al Manager di nome Notifier. Sfruttando un pattern di tipo *subscriber* il Notifier mantiene al suo interno le liste degli agenti interessati ai vari LSA, qualsiasi operazione realizzata sullo spazio viene comunicata anche a questo che a sua volta notifica a ogni sottoscrittore interessato tale avvenimento.

Alla struttura così delineata si affiancano componenti di contorno dalle funzionalità specifiche quali i DiffusionServer e DiffusionWorker a cui vengono demandate le operazioni di diffusione tra un nodo e l'altro, rispettivamente, in ingresso e in uscita in aggiunta a tutti i componenti che permettono ad entità esterni quali agenti di interagire con il nodo richiedendo operazioni sullo spazio.

2.3.2 LSA

La classe LSA è modellata tramite una stringa che funge da identificatore e un Content rappresentante il suo contenuto vero e proprio. Allo scopo di mantenere la maggiore generalità possibile e vista la gamma di possibili rappresentazioni utilizzabili per un LSA si è deciso di realizzare la classe Content come astratta e di fornire il sistema di una Factory atta a creare un LSA a partire da varie rappresentazioni o di passare da una all'altra. In particolare il sistema dà la possibilità di realizzare un LSA a partire da una rappresentazione Java o da una Prolog-like. Il Content a sua volta è formato da un insieme di coppie proprietà-valore, ciascuna di esse è un

singolo assegnamento per cui è definito il nome della proprietà stessa e il valore (o valori) ad essa assegnato rappresentato in ultima analisi da una stringa.

2.3.3 Reaction

Le eco-law sono modellate in questo prototipo attraverso la classe *Reaction* la cui rappresentazione astratta è riportata in figura 2.3.

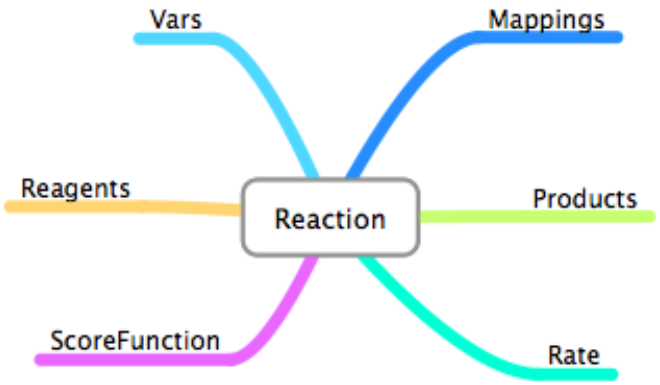

Figura 2.3: Struttura astratta di Reaction

Come è logico attendersi ogni reazione è formata da una lista di reagenti e prodotti. Mappe chiave-valore sono utilizzate per immagazzinare e accedere a tali elementi utilizzando come key-set il loro identificativo (che sappiamo essere univoco a livello di reazione).

Gli elementi *Vars* e *Mappings* sono invece strutture di supporto atte a gestire aspetti relativi all'esecuzione delle eco-law. In particolare Vars interessa l'assegnamento delle variabili, le quali essendo condivise tra reagenti e/o prodotti comportano obbligatoriamente che il loro scope sia definito a livello di reazione per permetterne l'accesso a tutti gli oggetti. Il concetto alla base della struttura Mapping è del tutto similare ma realizzato per mantenere gli assegnamenti tra identificatori degli LSA e identificativi di reagenti/prodotti poiché questi ultimi possono essere indicati come particolari variabili all'interno dei template stessi. In entrambi i casi anche qui vengono utilizzate delle mappe per velocizzare indicizzazione e recupero di tali elementi.

Infine i nodi ScoreFunction e Rate sono pensati per gestire gli aspetti relativi all'esecuzione della reazione secondo il modello CTMC. Il primo determina come il punteggio di matching dei singolo LSA reagenti determini la "bontà" della reazione mentre il secondo, facendo uso di tale informazione e del rate indicato in fase di creazione della reazione, determina quale sarà l'istante di tempo (a partire da quello corrente) in cui la reazione dovrà essere eseguita.

2.3.4 Componenti da riprogettare

Il prototipo è stato inizialmente sviluppato per un modello di eco-law con una sintassi e una espressività molto ridotta rispetto alla versione presentata in [11] e descritta approfonditamente al capitolo successivo su cui si basa questa tesi. Alla luce di ciò risulta chiaro che:

- la classe Reaction e tutte quelle concernenti la rappresentazione interna delle eco-law devono essere ripensate per realizzare l'espressività del nuovo modello;
- il ReagentMatcher sarà completamente rivisto alla luce delle modifiche di cui sopra per poter gestire l'aumento della complessità;
- quanto ai punti precedenti in parte si ripercuoterà sulla rappresentazione interna di dati e LSA comportando una parziale riprogettazione di questi;
- alcune strutture di boundary dovranno subire modifiche per rispecchiare nuove necessità sorte nella definizione del modello.

Capitolo 3

Modello e Linguaggio delle Eco-law

In questo capitolo si intende fare maggiore chiarezza sul modello delle eco-law, la sintassi del linguaggio e la relativa semantica. Il tutto viene inizialmente presentato, come normalmente viene fatto per i modelli di coordinazione, tramite un modello computazionale astratto corredato da un algebra di processo con semantica operazionale. In seguito si introduce un modello operazionale che è la base di partenza per la versione implementata all'interno del prototipo nel corso della tesi . La versione qui presentata è ad oggi la più recente ma non ancora definitiva del modello così come discussa in [11].

3.1 Modello computazionale astratto

3.1.1 Sintassi

Si assume che la meta-variabile i spazi sui possibili identificatori degli LSA, x su nomi liberi (per le variabili), p su nomi di proprietà, r su valori reali, l su letterali e n su identificatori di nodi. A ciò si aggiunga che data una meta-variabile b si intende con B l'insieme degli elementi di cui può assumere il valore e inoltre si indica con $\bar{b}$ una variabile che spazia su $\bar{B}$ che indica sequenze ordinate di B e con b_i si intende l'i-esimo elemento di tale lista. Infine si supponga che la meta-variabile θ spazi sulle sostituzioni di variabili con termini, indicato come $\{\mathsf{t}_1/\mathsf{x}_1,...,\mathsf{t}_n/\mathsf{x}_n\}$ dove i termini t_i possono essere

LSAs:			
Value	v	::=	$i \mid p \mid l \mid r$
Description	D	::=	$p{:=}(\overline{v}) \mid \overline{D}$
LSA	L	::=	$i\{D\}$
Filters:			
Variable	$\mathtt{x}$	::=	$?x \mid ?x.\mathrm{D}$
Term	t	::=	$\mathtt{x} \mid ?\{x : f\} \mid v \mid (\overline{v}) \mid \{D\}$
Operator	op	::=	`has` | `has-not` | =
Filter	F	::=	$t\ op\ t \mid$ `clones` $t \mid$ `extends` t
Ground Filter	G		(Filter with no variable)
Eco-laws:			
Pattern name	ν	::=	$i \mid ?x$
Pattern	P	::=	$\nu{:}[\overline{F}]$
Eco-law	E	::=	$\overline{P} \overset{t}{\mapsto} \overline{P}$
Molecule	M	::=	$L \mid P$
Transaction	T	::=	$\langle n : \overline{M} \overset{t}{\mapsto} \overline{M} \rangle$
Agents and LSA-space:			
Agent	A	::=	$0 \mid a_r.A \mid A + A \mid (A \mid A)$
Actions	a	::=	`inj`$(\nu, G) \mid$ `obs`(ν, F)
		|	`upd`$(\nu, G) \mid$ `rem`(ν)
Agent state	s	::=	$\langle n : (\bar{i}_o; \bar{i}_a)A \rangle$
Ecosystem	S	::=	$0 \mid L \mid E \mid T \mid s \mid n \overset{\overline{r}}{\rightsquigarrow} \overline{n} \mid (S \mid S)$

Figura 3.1: Sintassi dell'algebra di processo

di qualsiasi tipo sintattico e gli elementi $\mathtt{x}$ sono variabili usate nell'algebra. Una sostituzione θ applicata a un termine $\mathtt{t}$ è indicata con la notazione $\mathtt{t}\theta$.

La sintassi dell'algebra utilizzata è riportata in figura 3.1. Si osserva come un LSA è modellato come una coppia formata da un id i e una descrizione semantica D, a sua volta formata da un insieme di proprietà multivalore. Un filtro F può combinare due termini tramite l'operatore op o costituisce un filtro esteso tramite le funzioni `clones` o `extends` mentre si indicano i filtri ground, ovvero senza l'occorrenza di variabili, con la meta-variabile G. Un termine t è una variabile (del tipo $?x$ o con descrittore $?x.D$ usato per indicare la descrizione del LSA con id $?x$), un valore v, una lista di valori $(\overline{v})$ o una intera descrizione $\{D\}$.

Un pattern, o template, da utilizzarsi in una eco-law, è rappresentato tramite una coppia formata da un nome di pattern ν (a sua volta un id di LSA o una variabile) e una sequenza $\bar{F}$ di filtri. A sua volta l'eco-law è strutturata similmente a una reazione chimica e formata da una sequenza di

pattern nella sua parte sinistra-i template dei reagenti-e una sequenza sulla destra-i template dei prodotti-separate da un espressione rappresentante il rate t. Il processo secondo il quale una eco-law viene istanziata fino al divenire eseguibile è modellata tramite una transazione chimica dove ogni pattern definito nei reagenti viene sostituito con un LSA vero e proprio.

Un processo agente A è formalizzabile usando tecniche ispirate dalle algebre di processo, quali π-calculus o LINDA-calculus, e modella il comportamento nel tempo di un agente che opera inserendo/rimuovendo/osservando gli LSA. A è quindi definibile tramite il processo vuoto 0, l'azione prefissa a con un rateo r e seguita dalla continuazione A ($\alpha_r.A$), la scelta non deterministica di processi ("+") e la composizione parallela ("|"). Una azione comprende l'iniezione di un nuovo LSA con nome ν (che sarà istanziato con l'id del LSA) e contenuto descritto dal filtro ground G, osservazione del LSA ν con filtro F, modifica del LSA ν con filtro ground G e rimozione del LSA ν. Oltre al processo A, lo stato di un agente comprende anche il nodo n in cui è eseguito e due elenchi di id, quelli posseduti e quindi creati da esso ($\bar{i}_o$) e quelli accessibili e quindi collegati ad esso e solo osservabili ($\bar{i}_a$).

Infine, un ecosistema è un multiset composto da degli LSA (L), eco-law (E), transizioni (T), stati di agenti (s) e connessioni ($n \overset{\bar{r}}{\rightsquigarrow} \bar{n}$ che indicano la prossimità del nodo n con il vicinato $\bar{n}$ e con ogni elemento r_i che indica la distanza stimata tra n e n_i).

Oltre quanto detto si assume che gli operatori "+", "," e "|" siano commutativi, associativi e dotati della proprietà di assorbire $\bullet$(definiscono multiset) mentre "," sia anche idempotente così che `"1","2","1"` $\equiv$ `"1","2"` (definisce set).

3.1.2 Semantica operazionale

La semantica operazionale è definita come un sistema di transizioni etichettate con espressioni di rate del tipo $\langle \mathcal{S}, \longrightarrow, R \rangle$, in cui $\longrightarrow \subseteq \mathcal{S} \times R \times \mathcal{S}$: si scriverà $S \xrightarrow{r} S'$ per indicare che lo stato del sistema S può spostarsi in S' in seguito all'esecuzione di un evento schedulato con rate r. Nella definizione di $\longrightarrow$ si fa utilizzo di un altra transizione non etichettata tra gli stati del sistema $\hookrightarrow$, definita per modellare i passi intermedi dell'esecuzione di un eco-law. Con $\hookrightarrow^*$ si intende la sua chiusura transitiva e riflessiva.

Si fa uso anche della funzione parziale per l'applicazione dei filtri $\triangleright : \mathcal{G} \times \mathcal{D} \mapsto \mathcal{D}$, la quale prende un filtro F e una descrizione D e modella la manipolazione di D restituendo la versione aggiornata D'.

Le regole che descrivono la semantica dell'esecuzione delle eco-law sono indicate in figura 3.2.

$$\frac{(S' = n \overset{\overline{r}'}{\rightsquigarrow} \overline{n}') \text{ or } (S' = 0)}{n \overset{\overline{r}}{\rightsquigarrow} \overline{n} \,|\, S \xrightarrow{r} S \,|\, S'} \quad [\text{TOP}] \qquad \frac{\overline{L}|S \Vdash \langle n : \overline{L} \overset{t}{\mapsto} \overline{L}' \rangle}{\overline{L}|S \xrightarrow{r} \overline{L}'|S'} \quad [\text{MID}]$$

$$\frac{\langle n : E \rangle \,|\, S \hookrightarrow^* \langle n : \overline{L} \overset{r}{\mapsto} \overline{L}' \rangle \,|\, S' \quad \overline{L} \neq \overline{L}'}{E \,|\, n \overset{\overline{r}}{\rightsquigarrow} \overline{n} \,|\, S \xrightarrow{r} E \,|\, n \overset{\overline{r}}{\rightsquigarrow} \overline{n} \,|\, S'} \quad [\text{ECO}]$$

$$\frac{loc(D) = n \quad F\theta \triangleright D = D \quad \theta' = \theta \circ \{i/x, D/x.\text{D}\}}{\langle n : (x[F] + \overline{M} \overset{r}{\mapsto} \overline{M}') \rangle \,|\, S \,|\, i\{D\} \hookrightarrow \langle n : (i\{D\} + \overline{M} \overset{r}{\mapsto} \overline{M}')\theta' \rangle \,|\, S} \quad [\text{EREA}]$$

$$\frac{n \overset{\overline{r}}{\rightsquigarrow} \overline{n} \in S \quad F\theta \triangleright D = D' \quad loc(D') \in \overline{n}}{\langle n : (i\{D\} + \overline{L} \overset{r}{\mapsto} i[F] + \overline{M}) \rangle \,|\, S \hookrightarrow \langle n : (i\{D\} + \overline{L} \overset{r}{\mapsto} i\{D'\} + \overline{M})\theta \rangle \,|\, S \,|\, i\{D'\}} \quad [\text{EUPD}]$$

$$\frac{n \overset{\overline{r}}{\rightsquigarrow} \overline{n} \in S \quad F\theta \triangleright \delta(n, S) = D \quad loc(D) \in \overline{n} \quad fresh(i, S) \quad \theta' = \theta \circ \{i/x, D/x.\text{D}\}}{\langle n : (\overline{L} \overset{r}{\mapsto} x[F] + \overline{M}) \rangle \,|\, S \hookrightarrow \langle n : (\overline{L} \overset{r}{\mapsto} x\{D\} + \overline{M})\theta' \rangle \,|\, S \,|\, i\{D\}} \quad [\text{ENEW}]$$

Figura 3.2: Semantica operazionale per l'esecuzione delle eco-law

La regola [TOP] permette che le informazioni sui nodi vicini vengono modificate o cancellate in qualsiasi momento per riflettere la dinamicità del sistema. Conseguentemente [MID] si occupa di aggiornare lo stato degli LSA di contesto assumendo che il predicato $\Vdash$ (che modella un servizio dell'infrastruttura) osservi lo stato dell'ecosistema e istanzi una reazione da eseguire per realizzare qualsiasi aggiornamento di LSA necessario.

La regola [ECO] è responsabile per l'esecuzione di una intera eco-law. Si supponga E una eco-law presente nell'ecosistema e n un qualsiasi nodo, una reazione si costruisce e evolve fino ad essere pienamente istanziata in $\bar{L} \xrightarrow{r} \bar{L}'$ e porta lo stato del sistema in S'. Questo processo è modellato tramite $\hookrightarrow$, la cui semantica è esplicata nelle regole seguenti.

La regola [EREA] modella la selezione di un LSA reagente per un eco-law. Tramite il pattern $x : [F]$ nella parte sinistra seleziona un LSA $i\{D\}$ nello spazio la cui descrizione D faccia match con F (rappresentato da $F\theta \triangleright D = D$ per la sostituzione θ) e la sua location sia esattamente n; di conseguenza, definisce la sostituzione θ' con la composizione di θ e del mapping di x in i e di

$x.\text{D}$ nel contenuto di i, D. La configurazione risultante è ottenuta rimuovendo l' LSA selezionato dallo spazio e aggiornando la reazione corrente istanziando il pattern $x[F]$ con l' LSA $i\{D\}$, e applicando la sostituzione θ', necessario per propagare il risultato del match nei pattern che non sono ancora stati istanziati.

La regola [EUPD] si attiva solo quando la parte sinistra della reazione è completamente istanziata ovvero quando ogni pattern reagente è stato sostituito da un effettivo LSA. Tale regola ha l'obiettivo di aggiornare un LSA dello spazio come risultato di uno dei pattern prodotti nella parte destra della reazione. Sia $i[F]$ uno di questi pattern, e si assuma che $i\{D\}$ sia uno degli LSA reagenti selezionati che deve essere modificato dal pattern di cui sopra. Si applica $F\theta$ a D ottenendo la nuova descrizione semantica D', e si controlla che ancora appartenga al vicinato del nodo in cui l'eco-law è attivata. Come risulta si sostituisce il pattern $i[F]$ con l'LSA $i\{D'\}$ nella reazione, e lo si inserisce nello space: l'effettivo risultato è la sostituzione di $i\{D\}$ con $i\{D'\}$.

In ultimo la regola [ENEW] risulta simile a [EUPD] ma gestisce il caso in cui il pattern prodotto è utilizzato per generare un nuovo LSA, che è riconosciuto dal fatto che il pattern contenga una variabile x in luogo di un identificatore concreto i. La differenza principale è che si genera un identificatore nuovo i e l'LSA effettivo viene creato tramite l'applicazione del filtro F.

Le regole di transizione che descrivono il comportamento dell'ecosistema dovute a azioni eseguite da un agente sono presentate in figura 3.3. La regola [ASUM] gestisce la scelta non deterministica: se si possono eseguire i comportamenti A o A', se uno dei due può essere svolto allora si può escludere l'altro. La regola [APAR] è simile, con la sola differenza che il processo che non è selezionato per l'esecuzione di un azione non è escluso ma rimane nella composizione e può essere eseguito in un secondo tempo. Le regole [AINJ,AOBS,AUPD,AREM] sono strutturate per enfatizzare il loro legame con semplici eco-laws eseguite localmente. La regola [AINJ] è pensata per le operazioni di inject del tipo $\texttt{inj}(x, F)$ il cui effetto risulta equivalente a quello di un eco-law $(\bullet \overset{r}{\mapsto} x[F])$ che rappresenta l'inserimento di un nuovo LSA nello spazio senza esplicitare nessun reagente particolare, con sostituzione finale $\{i/x\}$ per l'identificatore nella continuazione dell'agente che rappresenta il risultato dell'operazione. La regola [AOBS] gestisce le operazioni di observe sugli LSA $\texttt{obs}(i, F)$ permesse solo se i è un id osservabile. La semantica è ottenuta tramite l'eco-law $(i[F] \overset{r}{\mapsto} i[F])$ che restituisce l'LSA iD, che fa match

$$\frac{\langle n : (\bar{i}_o; \bar{i}_a)A\rangle \mid S \xrightarrow{r} \langle n : (\bar{i}'_o; \bar{i}'_a)A''\rangle \mid S'}{\langle n : (\bar{i}_o; \bar{i}_a)A + A'\rangle \mid S \xrightarrow{r} \langle n : (\bar{i}'_o; \bar{i}'_a)A''\rangle \mid S'} \quad [\text{ASUM}]$$

$$\frac{\langle n : (\bar{i}_o; \bar{i}_a)A\rangle \mid S \xrightarrow{r} \langle n : (\bar{i}'_o; \bar{i}'_a)A''\rangle \mid S'}{\langle n : (\bar{i}_o; \bar{i}_a)A|A'\rangle \mid S \xrightarrow{r} \langle n : (\bar{i}'_o; \bar{i}'_a)A''|A'\rangle \mid S'} \quad [\text{APAR}]$$

$$\frac{\langle n : \bullet \overset{r}{\mapsto} x[G]\rangle \mid S \hookrightarrow^* \langle n : \bullet \overset{r}{\mapsto} i\{D\}\rangle \mid S'}{\langle n : (\bar{i}_o; \bar{i}_a)\mathtt{inj}(x, G)_r.A\rangle \mid S \xrightarrow{r} \langle n : (i, \bar{i}_o; i, \bar{i}_a)A\{i/x\}\rangle \mid S'} \quad [\text{AINJ}]$$

$$\frac{\langle n : i[F] \overset{r}{\mapsto} i[F]\rangle \mid S \hookrightarrow^* \langle n : i\{D\} \overset{r}{\mapsto} i\{F\theta \rhd D\}\rangle \mid S}{\langle n : (\bar{i}_o; i, \bar{i}_a)\mathtt{obs}(i, F)_r.A\rangle \mid S \xrightarrow{r} \langle n : (\bar{i}_o; i, \bar{i}_a, b(D))A\theta\rangle \mid S} \quad [\text{AOBS}]$$

$$\frac{\langle n : i[0_{\mathcal{F}}] \overset{r}{\mapsto} i[G]\rangle \mid S \hookrightarrow^* \langle n : i\{D\} \overset{r}{\mapsto} i\{D'\}\rangle \mid S'}{\langle n : (i, \bar{i}_o; \bar{i}_a)\mathtt{upd}(i, G)_r.A\rangle \mid S \xrightarrow{r} \langle n : (i, \bar{i}_o; \bar{i}_a)A\rangle \mid S'} \quad [\text{AUPD}]$$

$$\frac{\langle n : i[0_{\mathcal{F}}] \overset{r}{\mapsto} \bullet\rangle \mid S \hookrightarrow^* \langle n : i\{D\} \overset{r}{\mapsto} \bullet\rangle \mid S'}{\langle n : (i, \bar{i}_o; i, \bar{i}_a)\mathtt{rem}(i)_r.A\rangle \mid S \xrightarrow{r} \langle n : (\bar{i}_o; \bar{i}_a)A\rangle \mid S'} \quad [\text{AREM}]$$

Figura 3.3: Semantica operazionale per il comportamento di un agente

con il filtro F, e la relativa sostituzione θ da applicarsi alla continuazione dell'agente che anche in questo caso rappresenta il risultato dell'operazione. La regola [AUPD] è realizzata per le operazioni di update $\mathtt{upd}(i, F)$ (permesso solo se i fa parte degli id posseduti): il suo effetto sullo spazio è equivalente a quello dell'operazione $(i[0_{\mathcal{F}}] \overset{r}{\mapsto} i[F])$ (in cui con $0_{\mathcal{F}}$ si intende il filtro universale che fa match con qualsiasi descrizione) e senza nessuna continuazione da applicare. Infine la regola [AREM] gestisce operazioni di rimozione $\mathtt{rem}(i)$, attuabile solo se i è un id posseduto, e il suo effetto sullo spazio è lo stesso della eco-law $(\bullet \overset{r}{\mapsto} i[0_{\mathcal{F}}])$, senza continuazione da applicare e con la rimozione di i dagli insiemi di id posseduti e osservabili.

3.2 Modello operazionale

3.2.1 Descrizione informale

Quella qui presentata è una delle possibili concretizzazioni del modello astratto presentato nella sezione precedente, le scelte operate sono dipendenti da vari aspetti tra cui il compromesso tra espressività e semplicità del linguaggio e il riutilizzo di tecnologie esistenti per gestire e manipolare descrizioni. Alla luce di ciò le seguenti scelte sono state fatte:

1. Le eco-law devono essere *il più compatte possibile*. I filtri dei reagenti devono indicare le sole caratteristiche che un LSA deve possedere mentre quelli dei prodotti le sole da modificare.

2. I filtri devono *mantenere uno stile dichiarativo dove possibile*. Così come la loro sintassi/semantica deve rimanere omogenea se usata per reagenti e prodotti.

3. LSA e eco-law devono *essere di facile implementazione*. Ciò per evitare di aumentare troppo la complessità del middleware di SAPERE.

LSA

Gli LSA sono rappresentati secondo una notazione RDF-like (si veda [1]), probabilmente la tecnica più diffusa per rappresentare informazioni strutturate nel Web. La descrizione semantica di un LSA è strutturata come un insieme di proprietà multivalore o può equivalentemente essere vista come un insieme di coppie formate da un nome di proprietà e un valore. I valori utilizzabili sono letterali (stringhe), numeri, identificatori di LSA e nomi di proprietà. Un esempio di notazione astratta di un LSA è la seguente:

```
lsa1432{
  #location := ("room131"), #time := (1011233120),
  type := (person), age := (20), interests := (music,sport,travels)
}
```

Il termine `lsa1432` è l'identificatore dell' LSA, la proprietà `#location` è assegnato al valore singolo `"room131"` (una stringa), `#time` a una rappresentazione numerica dell'istante di creazione, `type` viene in generale utilizzato per indicare il tipo di LSA (valori comuni possono essere `field`, `display`,

`user`, ecc.) e infine la proprietà `interests` ha un insieme di valori composto da `music`, `sport` e `travels`.

Le proprietà `#location` e `#time` sono proprietà sintetiche (o di contesto), gestite completamente dal middleware, che nello specifico tengono traccia della locazione corrente in cui è si trova LSA e dell'istante di tempo in cui è stato creato. Oltre a ciò si suppone che alcuni LSA sintetici siano sempre presenti in ogni nodo. Gli esempi più importanti a riguardo sono rappresentati dal *LSA time* e dagli *LSA neighbour*. Il primo rappresenta il tempo locale del nodo mentre i restanti indicano le informazioni relative a ciascun vicino come si vede nell'esempio seguente:

```
lsa0232{
  #location := ("room131"), distance := (51.3), orientation := (north-east),
  location := ("room132")
}
```

Il nodo così descritto sarà alla locazione `"room132"` che si trova a una distanza (stimata) di 51.3 metri in direzione nord-est.

Rate

Come riportato anche al capitolo 1 per modellare lo scheduling delle eco-law si intende usare il framework delle Catene di Markov Tempo-Continue (CTMC). Tale decisione deriva principalmente dal fatto che le CTMC (*i*) riescono a modellare esattamente il comportamento chimico stocastico, (*ii*) sono molto frequenti in natura anche in ambiti al di fuori da quello chimico, (*iii*) sono ben studiate e documentate a livello di ricerca.

Nonostante ciò, quasi da subito, ci si è accorti che alcune eco-law dovrebbero essere eseguite con una semantica di tipo "il prima possibile", che è modellabile in CTMC tramite un rate infinito (o molto grande in rapporto agli altri). Nel proseguo eco-law di questo tipo saranno rappresentate senza l'indicazione del rate.

Eco-law

Il modello astratto di cui sopra definisce per le eco-law una struttura del tipo $P_1 + ... + P_n \xrightarrow{r} P'_1 + ... + P'_m$ in cui un pattern P è del tipo $x[F]$, con x nome del pattern e F un filtro. Si ricordi che un filtro di un reagente fa match con una descrizione semantica se la sua applicazione a essa non causa

cambiamenti, mentre nel caso di pattern di prodotti manipola la descrizione di un LSA nuovo o esistente.

Nella rappresentazione proposta un filtro intero è realizzato tramite una sequenza di filtri separati da ";", da applicarsi da sinistra a destra per manipolare una descrizione. Il caso di un pattern senza filtri del tipo $x[]$ sarà per brevità indicato con solo x. Il filtro `clones x.D` prende una descrizione e la converte nella descrizione del LSA che farà match con `x`. Nel caso di pattern di prodotto è utilizzato per copiare la descrizione di un LSA in un altro mentre nel caso di pattern reagente è utilizzato per determinare se due LSA hanno la stessa descrizione. Il filtro `extends x.D` è simile e prende una descrizione e vi aggiunge tutti gli assegnamenti proprietà-valore della descrizione del LSA che farà match con `x`. Nei pattern prodotto è utilizzato per "aggiungere" la descrizione di un LSA ad un altro, nel caso di pattern reagente invece è utilizzato per controllare se un LSA possiede almeno tutti gli assegnamenti presenti in un altro.

Altri filtri impongono i valori associati a una proprietà in maniera puntuale prendendo il nome di una proprietà nella parte sinistra e una lista a destra. Il caso `p=(v1,...,vn)` cambia i valori associati a `p` esattamente nei valori `v1,...,vn`: nei pattern reagenti pertanto ricerca descrizioni che possiedono questi valori associati a `p`, nei pattern prodotto cambia l'associazione di `p` a tali valori. In maniera simile, il caso `p has (v1,...,vn)` cambia i valori associati a `p` aggiungendo `v1,...,vn`, se alcuni sono già presenti non vengono ripetuti (stiamo utilizzando set). Infine il caso `p has-not (v1,...,vn)` cambia i valori di `p` rimuovendo `v1,...,vn`, se presenti.

Come precedentemente indicato, per permettere una maggiore generalità si intende aggiungere al linguaggio dei filtri la possibilità di utilizzare variabili di tipo logico al posto di termini e nello specifico da utilizzarsi al posto di proprietà, valori all'interno di liste o di intere liste. Saranno presenti due tipi di variabili quindi: (i) variabili non vincolate x da utilizzarsi al posto di qualsiasi termine (chiaramente una stessa variabile che ricorre più di una volta in una eco-law scatena una sostituzione che coinvolge tutte le sue occorrenze nello stesso modo) e (ii) e variabili annotate del tipo $\{x : f\}$ che possono essere usate per qualsiasi termine t tale che sostituendo x con t nella formula booleana f questa venga valutata positivamente. Un esempio di ciò è $\{x : x \text{ is } 1 + 5\}$ sostituibile solo con 6, $\{x : x < 3\}$ sostituibile con valori inferiori a 3 e $\{x : x \textit{ matches } y\}$ sostituibile con qualsiasi valore faccia match con y. Questa metodologia fa sì di poter incapsulare della

computazione esterna dentro le eco-law. A livello di modello operazionale si assume che siano supportate funzioni matematiche standard e funzionalità di matching semantico astratto. Un esempio in accordo con il modello proposto è realizzato dalla seguente eco-law:

Bind eco-law

```
S:[request has (BR); PR has-not (T)] +
BR:[type has (bind_request); matcher=(MT); bind_prop=(PR)]  +
T:[extends MT.D]
--->
S:[PR has (T)]  +  BR  +  T
```

L'eco-law richiede la combinazione di 3 LSA differenti: `S` è la sorgente di richiesta del collegamento, `T` è l' LSA target da collegare e `BR` è un LSA di richiesta, collegato alla sorgente e contenente tutti gli elementi della richiesta. I filtri della eco-law indicano che `S` è collegata a `BR` ma non è ancora collegata al target `T`; `BR` è di tipo `bind_request`, ha un matcher `MT` (che è l'identificatore di un altro LSA contenente le associazioni proprietà-valore che richiediamo al target), e specifica `PR` come proprietà in cui creare il collegamento. Per quanto riguarda `T` deve semplicemente estendere la descrizione di `MT`. Il risultato dell'applicazione di questa eco-law, da eseguirsi "il prima possibile", è la creazione di un collegamento da `S` a `T`.

3.2.2 Descrizione formale

Sistema di descrizione semantica

Lo si descrive tramite la tupla $\langle \mathcal{D}, 0_{\mathcal{D}}, loc(.), b(.) \rangle$ in cui $\mathcal{D}$ è il set delle descrizioni semantiche degli LSA, $0_{\mathcal{D}} \in \mathcal{D}$ è la descrizione vuota, $loc(.)$: $\mathcal{D} \mapsto N$ è la funzione di location che assegna a una descrizione D di un LSA l'identificatore del nodo n in cui è collocato e $b(.) : \mathcal{D} \mapsto \bar{I}$ è la funzione di bond ovvero di collegamento che assegna alla descrizione D il set di identificatori degli LSA a cui è al momento collegato. $\mathcal{D}$ e $0_{\mathcal{D}}$ sono definiti come segue:

Semantic descriptions:

Values	v	::=	$i \mid p \mid n \mid l$
Empty description	$0_{\mathcal{D}}$	::=	$\bullet$
Semantic description	D	::=	$p{:=}(\overline{v}) \mid \overline{D}$

La funzione $loc(.)$ semplicemente restituisce l'unico valore assegnato alla proprietà `#location` mentre $b(.)$ restituisce ogni LSA id utilizzato come valore di un assegnamento. Sono definiti dalle seguenti equazioni:

$$
\begin{array}{rcl}
loc(D, \texttt{\#location :=}(n)) & = & n \\
 & & \\
b(\bullet) & = & \bullet \\
b(p\texttt{:=}(\overline{v}), D) & = & b(\overline{v}), b(D) \\
b(v, \overline{v}) & = & b(v), b(\overline{v}) \\
b(i) & = & i \\
b(v) & = & \bullet \quad \text{otherwise}
\end{array}
$$

Sistema di pattern semantici

Lo si descrive tramite la tupla $\langle \mathcal{F}, 0_{\mathcal{F}}, .\triangleright. \rangle$, dove $\mathcal{F}$ è il set dei filtri semantici, $0_{\mathcal{F}}$ è il filtro universale che fa match con qualsiasi descrizione, e $.\triangleright. : \mathcal{G} \times \mathcal{D} \mapsto \mathcal{D}$ è la funzione di applicazione di filtro, una funzione parziale che preso un filtro F e una descrizione D, manipola quest'ultima fornendo una versione aggiornata D'. La sintassi di $\mathcal{F}$ è definita:

Eco-laws:

Formula	f		(A truth-valued expression based on terms t)
Term	t	$::=$	$x \mid x.\texttt{D} \mid \{x : f\} \mid v \mid (\overline{t_{\texttt{v}}})$
Operator	op	$::=$	`has` $\mid$ `hasnot` $\mid$ `=`
Universal Filter	$0_{\mathcal{F}}$	$::=$	$\bullet$
Filter	F	$::=$	$t_{\texttt{p}}\ op\ t_1 \mid$ `clones` $t_{\texttt{d}} \mid$ `extends` $t_{\texttt{d}} \mid \overline{F}$

Si assume un elemento F sia del tipo $F_1; ...; F_n$ (con ";" associativo e con la proprietà di assorbire $0_{\mathcal{F}}$), e t sia $t_1, ..., t_n$ (con "," associativo e commutativo). La sintassi utilizzata modella aspetti anche run-time del modello computazionale e risulta meno restrittiva di quanto richiesto. Le meta variabili per i termini annotate hanno il seguente significato: t_p può essere sia una proprietà che una variabile (anche annotata) che mantiene il nome della proprietà; t_l è una lista $(\overline{v})$ o una variabile (anche annotata) che rappresenta una lista; t_d è una variabile per una descrizione $x.\texttt{D}$ che sarà in seguito sostituita da una intera descrizione D; t_v è un valore o una variabile (anche

$$|\bullet| = \bullet \qquad |t, \bar{t}| = |t|, |\bar{t}| \qquad |v| = v \qquad |\{v : f\}| = v \text{ if } \epsilon(f) = true \qquad \text{[ERA]}$$

$$F; F' \rhd D = F' \rhd (F \rhd D) \qquad \bullet \rhd D = D \qquad \text{[SEQ]}$$

$$\begin{array}{rcll}
t_p\texttt{=}(\bar{t}) \rhd (|t_p|\texttt{:=}(\bar{v}), D) & = & |t_p|\texttt{:=}(|\bar{t}|), D & \text{[OP=]} \\
t_p \texttt{ has } () \rhd D & = & D & \text{[OP+a]} \\
t_p \texttt{ has } (t, \bar{t}) \rhd (|t_p|\texttt{:=}(|t|, \bar{v}), D) & = & t_p \texttt{ has } (\bar{t}) \rhd (|t_p|\texttt{:=}(|t|, \bar{v}), D) & \text{[OP+b]} \\
t_p \texttt{ has } (t, \bar{t}) \rhd (|t_p|\texttt{:=}(\bar{v}), D) & = & t_p \texttt{ has } (\bar{t}) \rhd (|t_p|\texttt{:=}(|t|, \bar{v}), D) \quad \text{if } |t|, \bar{v} \neq \bar{v} & \text{[OP+c]} \\
t_p \texttt{ has-not } () \rhd D & = & D & \text{[OP-a]} \\
t_p \texttt{ has-not } (t, \bar{t}) \rhd (|t_p|\texttt{:=}(|t|, \bar{v}), D) & = & t_p \texttt{ has-not } (\bar{t}) \rhd (|t_p|\texttt{:=}(\bar{v}), D) & \text{[OP-b]} \\
t_p \texttt{ has-not } (t, \bar{t}) \rhd (|t_p|\texttt{:=}(\bar{v}), D) & = & t_p \texttt{ has-not } (\bar{t}) \rhd (|t_p|\texttt{:=}(\bar{v}), D) \quad \text{if } |t|, \bar{v} \neq \bar{v} & \text{[OP-c]} \\
\texttt{extends } \bullet \rhd D & = & D & \text{[OPEa]} \\
(\texttt{extends } p\texttt{:=}(\bar{v}), D') \rhd (p\texttt{:=}(\bar{v}'), D) & = & \texttt{extends } D' \rhd (p\texttt{:=}(\bar{v}, \bar{v}'), D) & \text{[OPEb]} \\
\texttt{clones } \bullet \rhd D & = & D & \text{[OPCa]} \\
(\texttt{clones } p\texttt{:=}(\bar{v}), D') \rhd (p\texttt{:=}(\bar{v}'), D) & = & \texttt{clones } D' \rhd (p\texttt{:=}(\bar{v}), D) & \text{[OPCb]}
\end{array}$$

Figura 3.4: Semantica operazionale per l'applicazione dei filtri

annotata) che sostituisce un valore. Inoltre per la interpretazione e valutazione delle formule specificate nelle variabili annotate si assume l'esistenza di una funzione ϵ che presa una formula f restituisca un valore booleano.

La funzione di applicazione di filtro utilizza una funzione di cancellazione $|.|$che trasforma i termini in valori. La definizione di entrambe è riportate nelle equazioni di figura 3.4.

L'equazione [ERA] definisce la semantica della cancellazione la quale si applica elemento per elemento a liste di termini, trasformando valori in valori e convertendo variabili annotate nel valore istanziato v solo se la funzione f viene valutata positivamente. Si noti che la cancellazione è definita solo per termini ground.

L'equazione [SEQ] indica che la funzione di filtro applica i filtri da sinistra a destra fino al raggiungimento del filtro vuoto, che chiaramente non modifica in nessun modo la descrizione D

Le equazioni [OP*] definiscono la semantica di ogni tipo di filtro descritto in 3.2.1. Si noti che ad ogni termine a sinistra o destra di un operatore viene applicata la funzione di cancellazione prima di incidere su una descrizione.

Sistema di contestualizzazione

Lo si descrive tramite la tupla $\langle \delta(.,.), \Vdash \rangle$. $\delta(.) \in N \times \mathcal{S} \mapsto \mathcal{D}$ è la funzione di descrizione scheletro usata per generare una versione iniziale della descrizione di un nuovo LSA da creare e inizializzarne le proprietà sintetiche. $\Vdash \subseteq \mathcal{S} \times \mathcal{R}$ è invece il predicato per la gestione degli LSA sintetici che associa a uno stato di sistema S l'intera reazione R da eseguire per aggiornare lo stato degli LSA sintetici presenti.

Si definisce *time(n)* come il tempo attuale del nodo n. La funzione δ è definita come segue con l'intenzione di indicare che le uniche proprietà sintetiche considerate sono locazione e istante di creazione.

$$\delta(n, S) = (\texttt{\#location:=}(n), \texttt{\#time:=}(time(n)))$$

Per quanto riguarda $\Vdash$ per convenienza si introducono i seguenti LSA:

$$\begin{aligned} LSA\text{-}time(x,n,t) &\equiv x\{\texttt{\#location:=}(n);\ \texttt{type:=(\#time)};\ \texttt{\#time:=}(t)\} \\ LSA\text{-}nbr(x,n,d,n') &\equiv x\{\texttt{\#location:=}(n);\ \texttt{type:=(\#neighbour)}; \\ &\qquad \texttt{distance:=}(d); \texttt{location:=}(n')\} \end{aligned}$$

Alla luce di ciò si definisce "$\Vdash$" come segue:

$$\begin{aligned} S \mid (n \overset{\overline{r}}{\rightsquigarrow} \overline{n}) &\Vdash \langle \{\} : LSA\text{-}time(x,n,t) \mapsto LSA\text{-}time(x,n,time(n)) \rangle \\ S &\Vdash \langle \{\} : LSA\text{-}time(x,n,t) \mapsto \bullet \rangle \quad \text{if } (n \overset{\overline{r}}{\rightsquigarrow} \overline{n}) \notin S \\ S \mid (n \overset{\overline{r}}{\rightsquigarrow} \overline{n}) &\Vdash \langle \{\} : \bullet \mapsto LSA\text{-}time(x,n,t) \rangle \quad \text{if } LSA\text{-}time(x',n,t') \notin S \\ S \mid (n \overset{\overline{r}}{\rightsquigarrow} \overline{n}) &\Vdash \langle \{\} : LSA\text{-}nbr(x,n,d,n_i) \mapsto LSA\text{-}nbr(x,n,r_i,n_i) \rangle \\ S \mid (n \overset{\overline{r}}{\rightsquigarrow} \overline{n}) &\Vdash \langle \{\} : LSA\text{-}nbr(x,n,d,n') \mapsto \bullet \rangle \quad \text{if } n' \notin \overline{n} \\ S \mid (n \overset{\overline{r}}{\rightsquigarrow} \overline{n}) &\Vdash \langle \{\} : \bullet \mapsto LSA\text{-}nbr(x,n,r_i,n_i) \rangle \quad \text{if } LSA\text{-}nbr(x,n,d',n_i) \notin S \end{aligned}$$

In ordine procede a: aggiornare l' LSA time al cambiamento di stato, rimuovere tale LSA se il nodo scompare, creare l'LSA se un nuovo nodo appare e similmente a queste modifica, rimuove e inserisce un LSA neighbour.

3.3 Esempi di eco-law

In questa sezione si riportano alcuni esempi di eco-law per dare maggiore chiarezza alla sintassi e semantica operazionale. Ogni reazione sarà seguita da una breve spiegazione che mette in luce il comportamento che si vuole ottenere e gli aspetti particolari della sua realizzazione.

```
[PUMP]: Creates a field from an LSA
?LSA:[req_field_with_range=(?RNG); #location=(?L); pump_rate = (?RATE)]  +
?TIME:[type has (#time); time=(?T)]
--?RATE-->
?LSA  +  ?TIME  +  ?SRC:[extends ?LSA.D; range=(?RNG); pump_time=(?T); distance=(0);
                         type has (field, source); req_field_with_range has-not (?RNG);
                         source=(?LSA); source_loc=(?L); prev_ann=(?LSA); prev_loc=(?L)]
```

Questa eco-law è pensata per generare un campo ovvero una LSA che sarà successivamente diffusa (dalla eco-law seguente) per raggiungere un certo numero di nodi dell'ecosistema. Tramite l'utilizzo di campi si possono realizzare varie tipologie di interazioni tra cui probabilmente uno dei più interessanti è lo steering. La reazione per essere attivata necessità di due reagenti: il primo rappresenta l'LSA che vuole generare il campo e il secondo è l'LSA sintetico *time* che sappiamo essere sempre presente su ogni nodo. In particolare il reagente LSA deve indicare nella proprietà `req_field_with_range` un singolo valore che verrà assegnato alla variabile contrassegnata da `RNG` e il rate con cui si dovrà attivare questa reazione nella proprietà `pump_rate`. Per quanto riguarda il vincolo su `#location` siamo certi che sia sempre rispettato poiché tale proprietà statica è assegnata autonomamente dal sistema a tutti gli LSA presenti. Qualora le proprietà soddisfino i requisiti indicati l'eco-law sarà attivabile modificando lo spazio come indicato dai prodotti. I due LSA reagenti non verranno modificati in alcun modo poiché i rispettivi prodotti non contengono filtri, ne verrà invece creata una nuova annotazione come estensione di `LSA.D` e le modifiche indicate nei filtri successivi. In particolare risulta interessante come `range` venga settato al valore `RNG` che come dice il nome stesso vuole indicare il raggio entro il quale diffondere il campo, mentre `distance` viene posta a 0 ad indicare la distanza dal luogo di nascita del campo. Quale che fosse il tipo dell'LSA di padre all'estensione viene aggiunto, chiaramente, il tipo `field`. Si noti anche in questo caso come tramite una delle proprietà dei reagenti si vada a modificare la frequenza di esecuzione delle eco-law agendo sul rate della stessa.

```
[DIFF]: A field diffuses a cloned version in a neighbouring space
?FIELD:[type has (field); distance=(?D); #location=(?L); range=(?R); diff_rate=(?RATE)] +
?NEIGH:[type has (neighbour); neighbour_location=(?L1), distance=(?{D1: ?D1<?R-?D})]
?TIME:[type has (#time); time=(?T)]
--?RATE-->
?FIELD + ?NEIGH + ?TIME
?CLONE:[clones ?FIELD.D; distance=(?{D2: D2 is ?D+?D1}), prev_ann=(?FIELD);
        prev_loc=(?L); #location=(?L1); diff_time=(?T)]
```

Questa eco-law diffonde la copia di un LSA di tipo `field` come quello creato dalla precedente nei nodi vicini. Perché ciò sia possibile occorre che sia presente almeno un LSA `neighbour` che rappresenta l'esistenza di un nodo vicino indicandone anche la distanza relativa e location. Qui si fa uso di una variabile annotata sulla proprietà `distance` poiché occorre che tale valore risulti inferiore alla differenza tra il `range` (la distanza massima percorribile) e la distanza già percorsa. Se ciò risulta vero significa che il vicino si trova entro la portata del campo e quindi un LSA vi deve essere diffuso. L'unico effettivo prodotto della reazione è quello di creare una copia di `FIELD` dove in particolare la `#location` viene settata a quella del nodo vicino e la distanza aumentata della distanza a cui questo si trova. Il set della proprietà sintetica `#location` farà si che il sistema diffonda direttamente l'LSA nel nodo di destinazione indicato.

```
[NEWEST]: Of two fields, we keep the one with newest information
?FIELD:[type has (field); source=(?L); pump_time=(?PT)] +
?FIELD2:[type has (field); source=(?L); pump_time=(?{PT2: ?PT2 < ?PT})] --->
?FIELD
```

La precedente eco-law continuerà periodicamente ad attivarsi finché un campo sarà presente nel nodo. Dopo alcune interazioni ciò può portare alla duplicazione di un campo proveniente dalla stessa sorgente nei nodi adiacenti. L'eco-law di cui sopra risolve tale problema e nel caso due campi provenienti dalla stessa sorgente esistano nel nodo mantiene quello con `pump_time` più elevato poiché lo si suppone contenere delle informazioni più aggiornate. Del tutto similmente, essendo i nodi collegati tra loro può accadere che allo stesso nodo arrivino copie diverse dello stesso campo che hanno seguito percorsi diversi. In questo caso la discriminante da utilizzare sarà la distanza.

```
[BOND-PV]: Bond based on a property-value combination on target
?TARGET:[?PROP has (?VALUE)]  +
?SRC:[bond_request has (?BOND-REQ)]   +
?BOND-REQ:[type has (request_pv); bond_prop=(?B);
          target_prop=(?PROP); target_value=?VALUE]
--->
 ?SRC:[?B has (?TARGET)]  +  ?BOND-REQ  +  ?TARGET
```

In ultimo si presenta una ecolaw che fa uso di variabili in luogo di nomi di proprietà per ottenere l'instaurazione di un legame tra due LSA. Per fare ciò occorre che un LSA (`SRC`) abbia generato una `bond_request` che specifica le caratteristiche del LSA a cui vuole essere collegato. In questo caso le caratteristiche vengono definite sulla base di un legame proprietà-valore ovvero viene indicato il nome della proprietà che l'LSA obiettivo deve possedere (`PROP`) e parte dei valori che questa deve possedere (`VALUE`). Se ciò risulta verificato da un LSA che quindi sarà scelto come reagente per `TARGET` l'eco-law si attiverà collegando quest'ultimo a `SRC` e nello specifico aggiungendo il suo valore alla proprietà il cui nome è specificato in `bond_property`. Eco-law di questa tipologia sono molto importanti e utilizzate qualora si vogliano realizzare comportamenti basati sul matching e sul collegamento di LSA simili.

Capitolo 4

Sviluppo dei nuovi componenti

In questo capitolo si esporranno le scelte di progettazione per i nuovi componenti. Lo sviluppo riguarderà prevalentemente il framework delle eco-law cercando di lasciare immutate le parti restanti ove possibile. Le nuove entità da sviluppare dovranno sottostare a molti vincoli, la maggior parte dovuti al modello a cui si fa riferimento (cfr. 3) ma anche in parte alle scelte progettuali svolte nel prototipo così da permettere l'interazione con i componenti già presenti. La progettazione del software dovrebbe essere un attività svolta a più fasi che alterna analisi e modellazione prima di pervenire alla fase implementativa. Un approccio del genere non è stato del tutto applicabile per via dei vincoli imposti dall'interazione con la parte funzionante del prototipo. Conseguentemente, parte delle considerazioni e valutazioni di questo capitolo concernono aspetti, inevitabilmente, di basso livello.

4.1 Da Reaction ad Ecolaw

Entità centrale dell'opera di riprogettazione svolta nella tesi è senza dubbio quella deputata a modellare le reazioni all'interno dell'ecosistema. Bisognerà realizzare entità e costrutti utilizzando i quali l'utente possa rappresentare, in maniera astratta, le reazioni con un espressività almeno pari a quella fornita dal modello del linguaggio e a ciò occorrerà aggiungere la logica in grado di interpretare giustamente tale rappresentazioni e permettere l'esatta esecuzione delle eco-law. La classe Reaction sviluppata nel prototipo sulla base del lavoro presente in [5] non è più sufficiente in funzione della maggiore

complessità acquisita dal modello delle reazioni di SAPERE e presentata al capitolo precedente.

Principali aspetti discordanti tra la precedente implementazione e l'attuale modello sono:

- una profonda diversificazione tra rappresentazione di reagenti e prodotti in luogo di un modello che vede i due elementi possedere la stessa sintassi e diversificarsi semanticamente per il solo posizionamento rispetto la freccia di reazione;
- all'interno dei pattern reagenti/prodotti non risulta possibile indicare più di un filtro per singola proprietà con conseguente perdita espressiva in fase di selezione dei pattern reagenti e in fase di descrizione delle modifiche nei pattern prodotti;
- modello e implementazione delle variabili non abbastanza potente per gestire tutte le possibili situazioni definite dalla nuova sintassi, con prevalentemente l'impossibilità di definire variabili su nomi di proprietà, valori singoli dei set e rate delle reazioni;
- inevitabile assenza di funzionalità aggiuntive previste in corso d'opera quali, per esempio, i filtri estesi `clones` e `extends`.

Ad una prima lettura gli aspetti più critici del modello sono quelli relativi l'utilizzo, assegnamento e sostituzione delle variabili (normali o annotate) , in particolare durante le fasi di matching successive alla scrittura della eco-law, durante le quali si determina quali LSA possono candidarsi a reagenti.

Alla luce del complesso modello da rappresentare e delle profonde relazioni tra gli elementi che lo compongono, la fase di progettazione delle entità che compongono le eco-law si è quanto più possibile rifatta al modello astratto presentato al capitolo 3, così da permettere un confronto uno a uno tra componenti realizzati e elementi del modello. Ciò è stato fatto anche per consentire a futuri utilizzatori del framework un veloce apprendimento della struttura delle classi a partire da una buona conoscenza del modello astratto.

4.1.1 Rappresentazione astratta di Eco-law

In figura 4.1 è rappresentata la struttura della nuova classe Ecolaw. Come è logico che sia, tale entità è formata da (liste di) reagenti e (liste di) prodotti.

Figura 4.1: Ecolaw astratta

La classe mette a disposizione metodi per aggiungere elementi dei due tipi così come la possibilità di recuperarli indicando il loro identificativo che sappiamo essere univoco all'interno delle due liste.

Gli altri nodi presenti in figura indicano tutte quelle strutture di supporto alla definizione/esecuzione della reazione. In particolare gli oggetti *Vars* e *UndefinedVars* hanno l'importante funzione di gestire i valori assegnati alle variabili definite nella eco-law. Ogni volta che una variabile di qualche tipo viene settata, in seguito alla candidatura di un LSA in luogo di reagente, viene anche inserita in una di queste strutture per essere accessibile ad altri reagenti e prodotti da cui potrebbe essere condivisa. Entrambe le strutture a questo scopo sono realizzate tramite delle mappe chiave-valore che utilizzano come keyset il nome stesso della variabile per favorire semplicità e velocità di recupero. A differenza di Vars che può contenere valori qualsiasi, la struttura UndefinedVars mantiene per ogni chiave delle liste di valori perché, come dice il nome stesso, è pensata per la gestione di variabili che rimangono indefinite dopo la prima fase di matching, solitamente conseguenza dell'utilizzo di queste ultime al posto di nomi di proprietà. Pertanto quando una variabile di questo tipo viene rilevata occorre annotare a parte l'elenco dei possibili valori a cui poterla assegnare nelle fasi successive, che è quello per cui questa struttura è pensata. Nel momento in cui il valore di una di queste variabili sarà definito o imposto (cosa che deve accadere obbligatoriamente per potere eseguire l'eco-law) essa viene rimossa da quelle indefinite e spostata in Vars.

I nodi *Mappings* e *CandidateLSA* hanno funzione simile a quanto appena visto : mantengono traccia delle corrispondenze tra reagenti ed LSA e permettono un veloce raffronto sugli assegnamenti di questi ultimi e un rapido accesso agli LSA selezionati. La mappa Mappings associa ad ogni

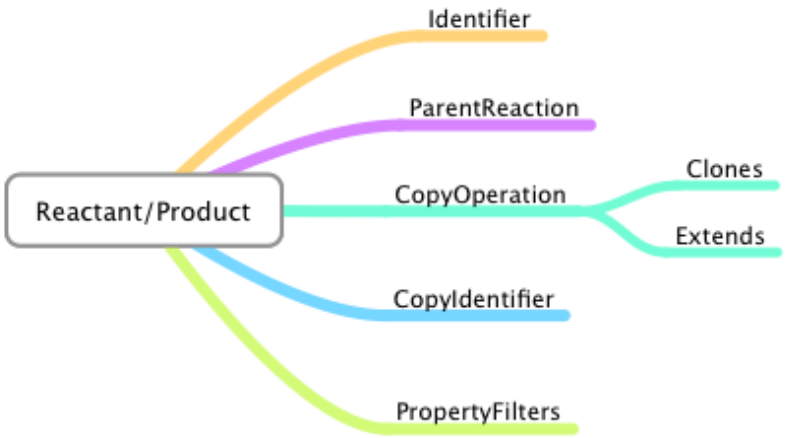

Figura 4.2: Struttura interna di Reactant e Product

identificatore di reagente l'identificatore dell' LSA selezionato in sua veste, la cui rappresentazione completa è mantenuta nella mappa CandidateLSA. Ciò risulta importante poiché tra i termini indicabili nei filtri dei pattern può essere usato l'identificatore di un reagente in vece dell'identificatore dell' LSA utilizzato, realizzando una sorta di variabile che però è direttamente dipendente dalla particolare scelta di candidati effettuata a runtime.

4.1.2 Reactant e Product

Questi componenti sono utilizzati in Ecolaw per modellare i pattern di selezione e modifica degli LSA dello spazio. Basandoci sul modello astratto, le due entità risultano strutturate nella stessa maniera, discendendo dalla classe padre *ChemicalTemplate*, e si differenziano solo per alcuni metodi che rendono disponibili all'utilizzatore. Uno schema della loro struttura è proposto in figura 4.2.

Ogni elemento viene riconosciuto e discriminato dagli altri sulla base di un identificativo. Gli identificativi sono univoci all'interno delle due liste e servono a determinare l'associazione tra reagente-prodotto poiché il secondo definirà le modifiche da apportare all'LSA scelto dal primo. Durante l'esecuzione l'identificativo verrà associato, attraverso Mapping, all'id dell'LSA

scelto come candidato e sarà accessibile come se si stesse usando una variabile. La corrispondenza univoca di identificatori tra reagenti e prodotti non è obbligatoria, infatti come visto al capitolo precedente, nelle Ecolaw la rimozione di LSA e la creazione di nuovi vengono semplicemente descritte non riportando nei prodotti l'identificatore del reagente da rimuovere nel primo caso, o inserendone uno nuovo (non utilizzato nei reagenti) nel secondo caso. Altro campo obbligatorio in fase di creazione di un pattern è il riferimento alla reazione per cui è definito, rappresentato da *ParentReaction*. Ciò permette ai metodi del template di accedere, attraverso l'eco-law padre, a tutte le strutture di supporto per le variabili definite sopra le quali hanno uno scope globale alla reazione.

CopyOperation e *CopyIdentifier* sono due elementi, la cui presenza è facoltativa, e sono utilizzati nella definizione dei filtri estesi. Infatti qualora si volesse definire un pattern come copia o estensione di un altro occorrerebbe indicare nel primo che tipo di filtro utilizzare (`clones` o `extends` rispettivamente) e nel secondo l'identificatore del *reagente* a cui l'operazione è riferita. Come già visto al capitolo precedente, pur utilizzando la stessa sintassi, il significato di tali filtri risulta chiaramente diverso a seconda che siano indicati in reagenti o prodotti. La definizione di un Reactant come clone di un altro reagente indica che i due LSA scelti come loro candidati devono essere necessariamente identici fatta eccezione per i filtri ulteriori indicati nel reagente clone. Il caso con estensione ammette invece che siano presenti nell' LSA scelto *almeno* tutti gli assegnamenti proprietà-valore presenti nell' LSA esteso. Chiaramente il rispetto di tali vincoli sono valutabili sono nell'ultima fase di matching quando ogni reagente ha un LSA candidato assegnato poiché equivalgono a filtri che creano una corrispondenza 1 a 1 tra due reagenti definendo variabili in comune su ogni proprietà e valore dei due LSA. Per quanto riguarda invece i pattern prodotti, l'indicazione di un filtro clone impone che l'LSA generato a partire da esso (che può essere anche nuovo) sarà in primo luogo una copia esatta dell' LSA a cui il CopyIdentifier fa riferimento con successiva applicazione dei restanti filtri. Analogamente a quanto appena visto, l'indicazione di estensione invece farà si che l'LSA assorba tutti gli assegnamenti proprietà-valore del suo target aggiungendoli ai propri.

Derivando dalla stesso padre, Reactant e Product possiedono metodi in comune per permettere, ad esempio, il recupero dei loro campi o dei filtri definiti al loro interno. Quello che li differenzia e caratterizza singolarmente sono nello specifico i metodi *filter* e *modify*. Il primo è presente in Reac-

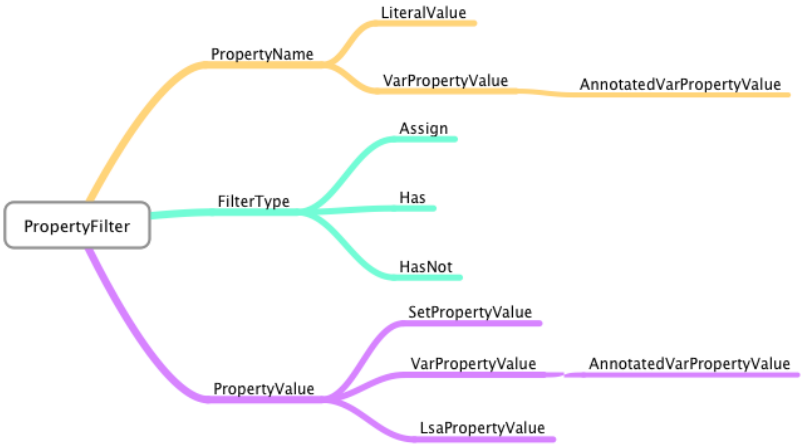

Figura 4.3: Struttura di un filtro su proprietà

tant e viene utilizzato durante la fase di matching in presenza di filtri estesi clones/extends poiché pensato per creare a runtime una copia (modificata come definito dagli altri filtri) del contenuto dell' LSA che si vuole clonare o estendere. Il secondo, *modify*, viene utilizzato per modificare un LSA sulla base dei filtri contenuti al suo interno da utilizzarsi all'atto di esecuzione dell'Ecolaw sullo spazio (cfr. 4.1.5).

PropertyFilter

Ogni pattern presenta al suo interno una lista di *PropertyFilter* che nel caso dei Reactant fungono da vincoli sulle proprietà degli LSA mentre per i Product specificano come le proprietà verranno modificate dall'esecuzione della eco-law. In figura 4.3 si riporta la struttura di un filtro comprensiva dei tipi di valori assegnabili ad ogni suo elemento interno. Il primo termine rappresenta il nome della proprietà su cui viene imposto il filtro ed è rappresentabile da un *LiteralValue* (un letterale) o da una variabile del tipo *VarProperrtyValue* o dalla sua sottoclasse *AnnotatedVarPropertyValue*. Nel primo caso avremo un filtro normale su una proprietà ben definita, nel secondo il filtro risulterà indefinito e quindi non valutabile fino all'assegnazione di un valore alla variabile.

Dopodiché un filtro è caratterizzato da un *FilterType*, una enumerazione

dei possibili filtri che possono essere `Assign`, `Has` e `HasNot`. Dipendentemente dal tipo di pattern, reagente o prodotto, in cui si utilizzano il loro significato è diverso: per i primi `Assign` impone un vincolo di uguaglianza, `Has` un vincolo di presenza di un elemento (o più d'uno) nella proprietà e `HasNot` il duale del precedente; per i secondi `Assign` si interpreta come un assegnamento completo (con rimpiazzo del valore precedente), `Has` nell'aggiunta di un valore e `HasNot` nella sua rimozione.

Infine si definisce il valore contenuto dal filtro e oggetto delle operazioni di match e modifica. Il tipo di questo elemento comprende *VarPropertyValue* e *AnnotatedVarPropertyValue*, *LSAPropertyValue* e *SetPropertyValue*. Le tipologie di variabili definibili sono due a rispecchiare quanto definito nel modello. La prima è una variabile non vincolata che può accettare un qualsiasi valore in suo luogo, la seconda, realizzata dalla sottoclasse *AnnotatedVarPropertyValue*, rappresenta le così dette variabili con annotazione. Queste sono rappresentate nella sintassi del linguaggio con `{x: f}` in cui la variabile x può solo accettare valori che sostituiti nel predicato booleano f lo verificano con esito positivo. La funzione f è del tipo (x *op exp*) ovvero presenta la variabile indicata, seguita da un operatore e poi da un'espressione. Come detto l'operatore deve essere di tipo booleano in modo che la sua valutazione ritorni un valore true/false, al momento sono supportati tutti gli operatori di confronto tra valori ma facilmente risulta possibile aggiungerne altri a scelta dell'utente. L'espressione con cui confrontare il valore per determinare se sia ammissibile può essere formata da un singolo valore ground, una variabile semplice (bindata a un altra proprietà nell'eco-law) o una vera e propria espressione con le suddette tipologie quali operandi. Nell'ultimo caso occorre sempre verificare che l'espressione sia ground prima di andare a compararla ovvero che sia valutabile e non vi siano variabili slegate. Per permettere una veloce prototipazione si è deciso di sfruttare una classe che utilizza l'engine Javascript per realizzare l'entità *Expression* con il risultato che gli unici operatori supportati al suo interno sono quelli standard matematici. Nonostante l'attuale implementazione risulti al momento sufficientemente espressiva per gli obiettivi preposti, qualora si volesse potenziare questo modello basterà solo ridefinire tale classe implementando il riconoscimento e utilizzo dei nuovi operatori che si ritengono necessari. Da come descritto, l'utilizzo di una variabile annotata può sembrare limitato ai soli pattern reagenti ma, in realtà, trova utilizzo anche nella parte destra della reazione: una delle operazioni definibili nella annotazione è `is`, che è interpretato come un assegnamento

della variabile al valore valutato nell'espressione. In questo modo è possibile iniettare un minimo di computazione all'interno delle eco-law e, come vedremo al capitolo successivo, questo meccanismo è spesso sfruttato in particolari tipologie di eco-law.

Il significato di LSAPropertyValue è quello di segnaposto per l'identificatore di un LSA specifico scelto come reagente a cui questo elemento si riferisce.

Il tipo LSAPropertyValue deve essere invece utilizzato quando in un filtro si vuole indicare l'identificatore di una degli LSA candidati a reagenti. Esso viene istanziato indicando il nome del reagente ma questo verrà usato come segnaposto per il valore effettivo dell'id del LSA che viene scelto in veste del reagente stesso. È quindi una sorta di variabile il cui assegnamento è eseguito automaticamente in fase di candidatura degli LSA a reagenti. Il suo funzionamento può sembrare simile a quello delle variabili, ma la logica che vi sta dietro è leggermente diversa. Per tale motivo e per evitare confusione questi due elementi si sono lasciati separati. L'utilizzo di tale tipologia di proprietà nei filtri è di grande importanza, poiché è tramite questi che le eco-law possono arrivare a collegare insieme diversi LSA, inserendo i valori dei rispettivi identificatori nelle proprietà degli altri.

Indicando un SetPropertyValue come oggetto del filtro si impone l'uguaglianza della proprietà a un set di valori che può contenere a sua volta elementi LiteralValue (ground), variabili o LSAPropertyValue.

Il numero di filtri definibile non è vincolato. Se nessuno viene indicato avremo un Reactant che farà match con qualsiasi LSA mentre nel caso di un Product non saranno eseguite modifiche all'LSA durante l'esecuzione della reazione. Possono essere definiti filtri sulla stessa proprietà in numero arbitrario. Ciò fornisce una ottima espressività agli utilizzatori a patto di usare oculatezza ed evitare di definire sulla stessa proprietà filtri discordanti.

Una volta creato il pattern tramite il passaggio dei filtri, questi ultimi vengono ordinati in maniera da snellire le operazioni di matching. Essendo infatti il matching eseguito sequenzialmente, confrontando un filtro dopo l'altro con le proprietà dell' LSA in esame, indicando per primi quelli maggiormente restrittivi si riconosceranno prima gli LSA che non soddisfano i vincoli con il risultato di snellire parte della computazione richiesta.

4.1.3 Rate

L'oggetto Rate è responsabile del calcolo della frequenza di esecuzione di ogni reazione. Le classi definite a questo scopo sono due: *MarkovianRate* e *ASAPRate*. Il primo viene utilizzato per modellare un evento di un sistema CTMC (come descritto in 1.3.1) e pertanto caratterizzato da un valore che stima la frequenza media di esecuzione. Questo valore può essergli fornito direttamente se specificato ma l'oggetto accetta anche variabili qualora si voglia controllare la velocità di reazione sulla base di alcune proprietà definite negli LSA in gioco. Quale che sia il tipo fornito, tramite il metodo *getNextOccurence* la classe fornirà il prossimo istante di tempo di avvenimento dell'evento a partire da un determinato istante passato come parametro di ingresso. Qualora si sia indicato un rate tramite una variabile, viene qui controllato che essa punti a un valore reale positivo; nel caso non sia così la reazione viene comunque schedulata ma il rate utilizzato è di 1.0. La classe *ASAPRate* viene invece utilizzata quando, in fase di definizione delle ecolaw, nessun rate di reazione viene indicato. Secondo la sintassi del linguaggio ciò vuole simulare l'utilizzo di un rate infinito per modellare una reazione la cui esecuzione deve essere immediata. In questo caso non si procede a calcoli matematici, ma semplicemente il metodo *getNextOccurence* restituisce in uscita lo stesso valore fornito in ingresso.

4.1.4 ScoreFunction

Lo score di una reazione è un'astrazione che rappresenta idealmente, con un valore numerico, il livello di corrispondenza riscontrato tra i reagenti e gli LSA scelti come candidati. Supponendo di poter selezionare i reagenti da un punto di vista semantico, quindi, potremmo dire che lo score indica la bontà degli LSA scelti e, conseguentemente, occorrerebbe che le reazioni migliori venissero eseguite prima delle altre. Per fare ciò si è realizzata la classe astratta *EcolawScoreFunction* che, sulla base del punteggio di match di ogni LSA con un reagente, calcola il punteggio totale relativo a una certa istanza di eco-law. Le funzioni definibili sono molteplici, e l'utente può crearne di nuove definendo una nuova sottoclasse e implementando il metodo astratto *getScore()*. Le versioni già implementate nel progetto determinano il punteggio in due maniere: attraverso una media dei punteggi di tutti i reagenti, o come prodotto di questi.

Quale che sia il metodo utilizzato per calcolare lo score, esso viene poi utilizzato nel calcolo delle attivazioni delle eco-law dalla classe Rate, che lo moltiplica per il rate statico fornitogli in creazione. Ciò fa sì che a valori più alti dello score corrispondano frequenze medie maggiori, con la possibilità di esecuzioni più frequenti. In particolare, questa funzionalità risulterà interessante da sviluppare e utilizzare quando, in versioni future del progetto, sarà supportato il match semantico dei reagenti così che effettivamente si possano avere per varie istanze di reazione score molto diversificati sulla base delle proprietà degli LSA usati.

4.1.5 Generazione della Transaction

Nel momento in cui una Ecolaw viene completamente istanziata dal ReagentMatcher (cfr. 4.2), oltre all'istante di tempo in cui schedularla, occorre determinare la lista di operazioni che, eseguite sullo spazio, realizzeranno l'effettiva trasformazione dei reagenti in prodotti. Ciò è possibile invocando sulla reazione completamente formata il metodo *getExecutableTransaction*, ottenendo in ritorno un elemento di tipo *Transaction*. Questo oggetto non è altro che una lista di operazioni concrete quali `inject`, `update` e `remove` da effettuarsi sullo spazio, generate a partire dai prodotti. Pur trattandosi di singole operazioni, in generale una per ogni prodotto definito, queste vengono raggruppate in un unico oggetto, poiché rappresentano l'effetto netto globale dell'esecuzione di una singola eco-law e, pertanto, devono essere tutte eseguite in sequenza e atomicamente. La generazione della transazione parte dal confronto tra la lista dei reagenti e la lista dei prodotti sulla base degli identificatori. Tutti gli identificatori di reagenti che trovano riscontro in quelli dei prodotti saranno riferiti a delle operazioni di update sugli LSA, ovvero delle semplici modifiche del loro contenuto. Diversamente, qualora vi fossero degli identificatori che compaiono solo in una delle due liste, avremo in un caso una remove di un reagente oppure una inject di un nuovo LSA. Nel caso si operi la creazione di un LSA, occorrerà anche richiedere allo spazio un nuovo identificatore, poiché potrebbe dover essere collegato a uno degli altri LSA prodotti. La logica per la generazione delle singole operazioni è integrata all'interno dei componenti che realizzano le relative entità astratte, ovvero i prodotti. In particolare nel caso di operazioni di update e inject, l'LSA da modificare o creare viene passato al metodo *modify* del rispettivo prodotto, il quale gli applica tutte le trasformazioni definite nei suoi filtri

interni così da restituire direttamente il nuovo LSA che sarà da aggiornare o inserire nello spazio.

Prima di essere passate al Manager, le Transaction subiscono una operazione di controllo alla ricerca di possibili operazioni di diffusione. Il vecchio modello del linguaggio delle eco-law prevedeva che le operazioni di diffusione tra nodi venissero indicate da una sintassi particolare all'interno dei prodotti. Diversamente, il nuovo modello definito realizza le diffusioni in maniera trasparente, basandosi sull'assegnamento agli LSA di valori diversi della proprietà sintetica `#location`. Pertanto è compito del sistema, prima di eseguire una transazione, controllare che le operazioni di inject contengano le location del nodo corrente, nel caso ciò non avvenga l'operazione richiesta è una diffusione e non va gestita normalmente ma tramite la logica deputata.

4.2 ReagentMatcher

Altro componente riprogettato interamente è il *ReagentMatcher*, il cui compito è quello di effettuare le associazioni tra i reagenti e gli LSA. È un componente passivo le cui funzionalità vengono invocate dal *ReactionManager* per la selezione dei candidati LSA a reagenti e il calcolo dello score delle reazioni. L'unico metodo pubblico del Matcher, *score*, accetta in ingresso un elenco di LSA, una reazione ed un livello di soglia con il quale filtrare i reagenti selezionati. Il metodo, eseguite varie fasi di matching che si descriveranno successivamente, ritornerà un elenco di score ad ognuno dei quali corrisponderà una reazione completamente istanziata comprensiva anche della *Transaction* da eseguire sullo spazio. Il motivo per cui viene ritornato un elenco di score è dovuto al fatto che, dato un pool abbastanza grande di LSA, potrebbe accadere che esistano più combinazioni diverse di LSA come reagenti che attivano la reazione.

La logica di funzionamento del metodo *score* è riportata nel diagramma di figura 4.4. Innanzitutto la lista dei reagenti viene recuperata ed è controllata la presenza di almeno un reagente. Infatti, nel caso non ve ne fossero, si avrebbe a che fare con una Ecolaw che presenta solo dei prodotti, ovvero che genera LSA a partire da nessun reagente. Qualora non sia questo il caso si procede a una prima fase di matching con lo scopo di determinare per ogni reagente tutti i possibili candidati dall'elenco di LSA passati. Trattandosi della prima fase di matching, solo i filtri che impongono vincoli molto restrit-

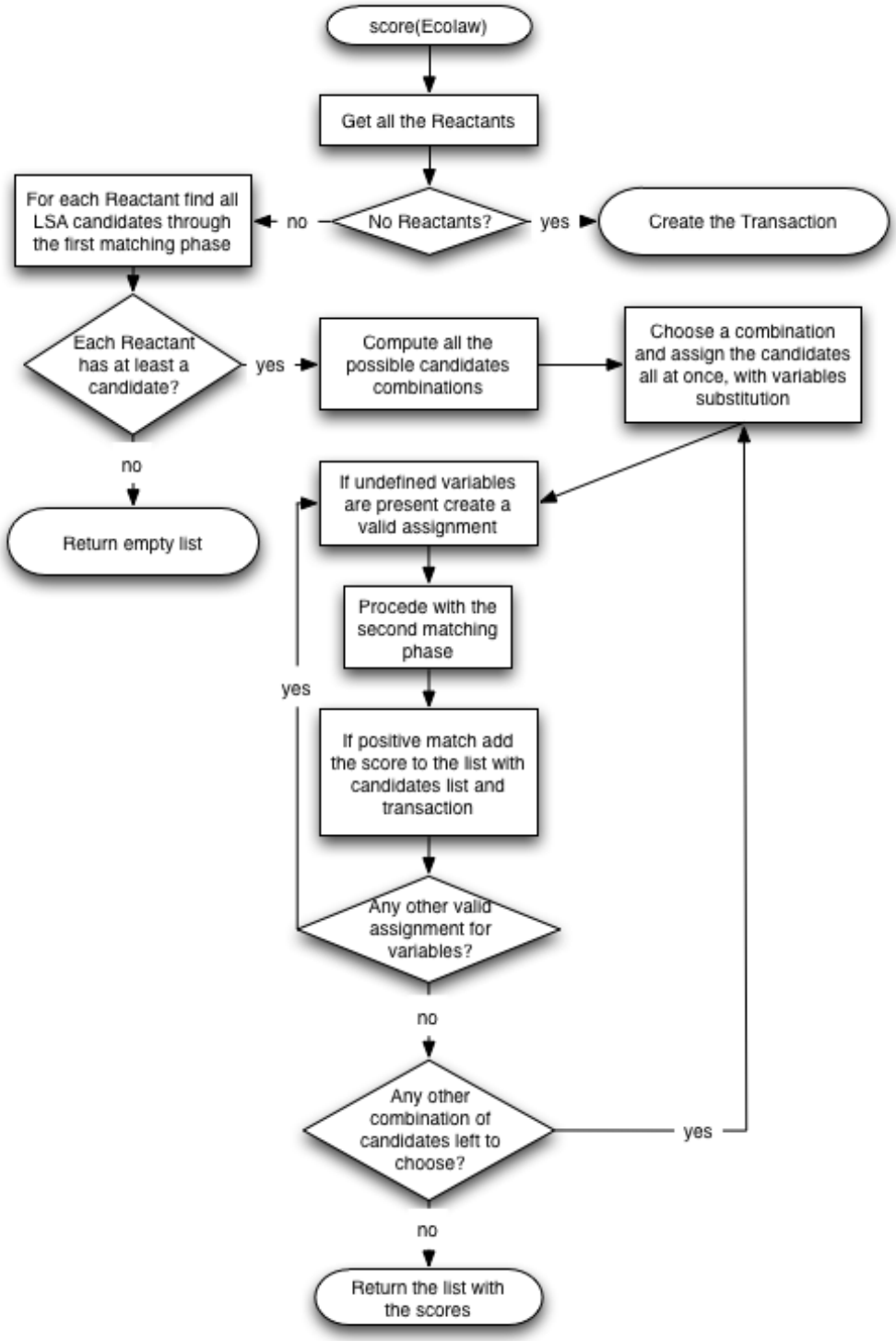

Figura 4.4: Diagramma di funzionamento del ReagentMatcher

tivi possono essere controllati ovvero quelli caratterizzati da valori ground. Infatti tutti i filtri basati su variabili non possono essere valutati non essendo nessuna di queste ancora stata assegnata. Dopo questa fase si ottengono per ogni reagente un elenco di LSA candidati. Nel caso in cui per un reagente non sia presente nessun LSA candidato, la reazione non potrà essere svolta e il metodo non ritornerà nulla. Nel caso invece sia presente almeno un candidato per ogni reagente, si procede a creare tutte le possibili combinazioni di LSA. Per ogni combinazione possibile si assegnano i candidati ad ogni reagente tutti in una volta, tale operazione è molto importante poiché è in questa fase che si popolano le strutture *Vars*, *UndefinedVars* e *Mapping* viste in precedenza. Nel caso di eco-law che fanno un uso molto spinto di variabili in luogo di nomi di proprietà può accadere che dopo questa fase risultino esserci variabili indefinite. Se ciò si verifica, in maniera similare a quando si creano le combinazioni di reagenti, vengono generati tutti gli assegnamenti possibili per ogni variabile rimasta slegata. Preso uno degli assegnamenti possibili si sostituisce ai valori ancora indefiniti e si procede alla seconda e ultima fase di match poiché a questo punto tutte le variabili sono assegnate così come anche gli LSA ai reagenti pertanto sia i *VarPropertyValue* e gli *LSAPropertyValue* sono valutabili. Se tutti gli LSA assegnati fanno match con i reagenti la reazione è attivabile, si valuta lo score e si crea la *Transaction* specifica per realizzarla sullo spazio. Si procede quindi a valutare una nuova configurazione di assegnamenti alle variabili indefinite se ancora presenti, altrimenti si procede con una combinazione di reagenti. Quando tutte le combinazioni vengono valutate il metodo ritorna fornendo al ReactionManager la lista di score che rappresenta le eco-law attivabili.

Le modalità di realizzazione del matching vero e proprio sono specificate nelle concretizzazioni dei metodi astratti *match()* e *fuzzyMatch()*. Il primo viene utilizzato per determinare il matching tra un singolo pattern reagente e un LSA. Quello che fa è scorrere uno per uno i filtri del pattern cercando corrispondenze all'interno del contenuto dell'LSA restituendo in uscita un punteggio di somiglianza. Questo stesso metodo viene anche utilizzato per il filtraggio iniziale dei candidati a reagenti, ma in questo caso senza tenere conto dei filtri che presentano variabili per i motivi già esposti precedentemente. Il secondo metodo, fuzzyMatch(), viene invece invocato dal primo qualora sia presente una variabile di tipo *AnnotatedVarPropertyValue* poiché è incapsulata in questo metodo la logica riferita ad ogni operatore utilizzabile al loro interno. Al momento la concretizzazione fornita gestisce gli operatori

di confronto standard ($<, \leq, >, \geq$ e $=$) con l'aggiunta di `matches`. Quest'ultimo controlla la corrispondenza di due letterali tra loro (che non possono essere convertiti in valori numerici) ma è stato inizialmente pensato per operazioni di match semantico che non fanno parte degli obiettivi di questa tesi ma rientreranno nei lavori futuri.

Altre modifiche minori sono state apportate a varie classi del progetto oltre alle suddette. Dal punto di vista del modello, per esempio, sono state riviste parte delle classi deputate alla rappresentazione degli LSA (contenuto e identificatore), così come sono state modificate alcune delle parti dell'engine del nodo per supportare l'inserimento e la gestione delle proprietà sintetiche degli LSA da parte del sistema. Non si espongono tali modifiche in dettaglio poiché prevalentemente determinate da necessità implementative e poiché rappresentano una parte poco rilevante del lavoro svolto nella tesi se confrontate al lavoro descritto precedentemente.

Capitolo 5

Casi di Studio

5.1 Personalizzazione del contenuto in base al contesto

L'idea alla base di questo caso di studio è che un display pubblico sia in grado di personalizzare il suo contenuto in accordo con le preferenze e gli interessi degli spettatori che ha davanti. Per fare ciò il display dovrà essere in grado di determinare la presenza di un utente e ottenere, conseguentemente, accesso al suo profilo. Una volta che si avrà accesso ai suoi interessi questi potranno essere confrontati con una serie di contenuti e risorse a disposizione del display che sceglierà quella più adatta da visualizzare. In aggiunta si vuole che il display riesca a gestire un numero arbitrariamente grande di utenti nelle sue vicinanze capendo quali contenuti riscontrano maggior consenso all'interno del gruppo. La decisione su quale contenuto si dovrà mostrare può, nel caso più semplice, essere realizzata con un semplice algoritmo di confronto su parole chiavi tra una lista di preferenze utente e i concetti con cui sono descritte le risorse. A quanto detto si vuole aggiungere la possibilità di far in modo che il sistema rilevi la presenza di utenti molto vicini posti di fronte allo schermo e non solo nei suoi paraggi. Nel caso ciò accada si prevede che una parte dello schermo possa indicare informazioni di interesse del solo utente senza interferire con la visualizzazione principale. In ultimo il sistema dovrà adattare i suoi contenuti qualora gli utenti modificassero "al volo" le proprie preferenze, compatibilmente con i contenuti a disposizione.

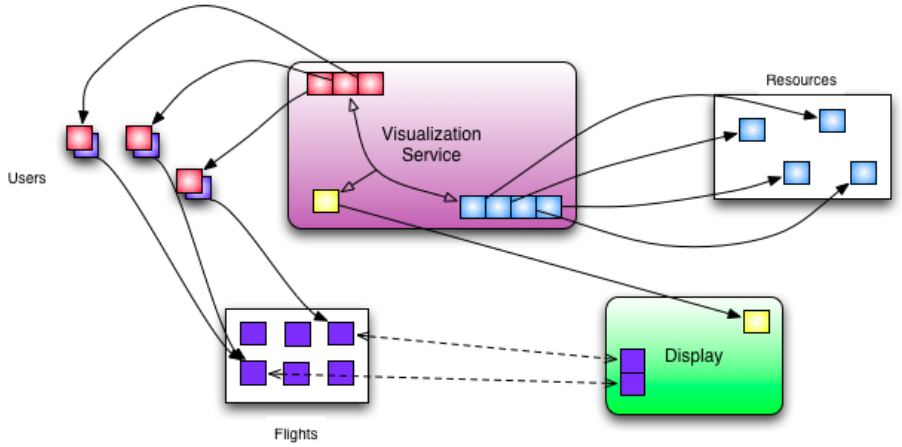

Figura 5.1: Architettura generale del primo caso di studio

5.1.1 Architettura dell'applicazione

In figura 5.1 si riporta uno schema concettuale dei componenti interessati che possono essere mappati sulle entità (ed LSA) del sistema che si vuole realizzare. Prima di tutto saranno presenti degli utenti dotati di profili contenenti le preferenze individuali. Ad ogni utente saranno anche legate le informazioni relative al suo posizionamento spaziale, così da determinarne la prossimità rispetto il display. I contenuti visualizzabili faranno a loro volta parte del sistema, ognuno di essi indicherà una risorsa vera e propria, localizzata presumibilmente sul nodo stesso o in rete, caratterizzata da un argomento a cui fa riferimento. L'entità preposta alla visualizzazione dei contenuti sarà il display su indicazione di un servizio separato, che costituisce la logica di determinazione del contenuto da visualizzare sulla base degli utenti di cui è a conoscenza.

5.1.2 Realizzazione dell'applicazione

L'applicazione per il caso di studio è stata realizzata tramite l'utilizzo di device fisici (smartphone) per rappresentare gli utenti e di entità software per realizzare il servizio di visualizzazione e per simulare il display. Di seguito

si descrive brevemente il comportamento di ogni entità, comprensivamente delle LSA che usa per interagire con gli altri componenti del sistema.

Contenuti

Queste entità rappresentano le risorse visualizzabili dal display. Si dividono in due tipologie: contenuti pubblicitari e informativi. I primi saranno quelli tra cui scegliere per la visualizzazione di gruppo mentre i secondi saranno utilizzati per la visualizzazione personalizzata per quegli utenti che si trovano vicino al display. I primi vengono realizzati tramite dei video che risiedono sul nodo e il cui percorso è presente negli LSA che li rappresenta mentre per i secondi sono previste solo informazioni statiche riportate direttamente all'interno del loro LSA. Nella demo si è deciso di pensare al sistema situato in un aeroporto, pertanto i contenuti informativi saranno riferiti a dei voli in partenza.

Gli LSA dei contenuti sono realizzate come segue:

Contenuti pubblicitari

- **type:** (`resource`);
- **topic:** l'argomento a cui fa riferimento il contenuto (video) in questione;
- **path:** il percorso da cui recuperare il contenuto.

Contenuti informativi

- **type:** (`flight`);
- **flight_number:** il codice del volo in questione;
- **info:** le informazioni da visualizzare relative a questo volo.

Users

Gli utenti sono gli spettatori del sistema e possono agire proattivamente solo in relazione alla modifica delle proprie informazioni personali. Durante la simulazione ogni utente viene realizzato utilizzando uno smartphone in grado di eseguire operazione sul LSA-Space del nodo. Oltre alla propria LSA che ne mantiene i dati tra cui le preferenze e il numero del volo interessato, ad

ogni utente è anche legata una seconda LSA che ne rappresenta la posizione in relazione al nodo SAPERE a cui è connesso e può essere utilizzata per determinare la prossimità.

Le proprietà principali dell'LSA utente sono:

- **type:** (`user`);
- **preferences:** gli interessi dell'utente;
- **flight:** il codice del volo di cui vuole ricevere informazioni;
- **position:** contiene l'id dell' LSA di posizione legata all'utente.

Le proprietà utilizzate dell'LSA posizione sono:

- **type:** (`localization`);
- **signal:** la potenza con cui il segnale del device dell'utente viene percepito dal sistema;

Display

Il display è una semplice entità reattiva che visualizza i contenuti che gli vengono collegati (video) e stampa messaggi informativi qualora un utente sia molto vicino.

Le proprietà presenti nel LSA display sono:

- **type:** (`display`);
- **visualization:** modalità di funzionamento, può essere `on` (in funzione) o `ready` (in attesa);
- **resource:** contiene l'id della risorsa da mostrare (video);
- **information:** contiene l'id delle informazioni da mostrare;

Servizio di visualizzazione

Questa entità realizza la logica con cui vengono selezionati i contenuti da mostrare. È l'unica entità che compie una minima computazione poiché incaricata di determinare tra un pool di risorse qual è la migliore da visualizzare sulla base delle preferenze degli utenti presenti nel sistema e ad esso collegati. Una volta che la decisione è stata presa il servizio modifica di conseguenza la propria LSA. Ciò sarà recepito da una eco-law che provvederà a mettere in funzione il display con il contenuto opportuno.

Le proprietà presenti nel LSA del servizio sono:

- **type:** (`visualization_service`);
- **users:** proprietà a cui vengono collegati gli LSA degli utenti;
- **available_resources:** proprietà a cui vengono collegati gli LSA delle risorse;
- **res_to_visualize:** contiene l'id della risorsa che il display dovrà mostrare;

5.1.3 Eco-law utilizzate

Di seguito si descriveranno brevemente le eco-law utilizzate nel caso di studio in esame per ottenere il comportamento atteso.

```
[BOND-US] Bonds a new user with preferences to a visualization service
?USER:[type=(user); preferences=?{PREF: ?PREF != ()}] +
?SERVICE:[type=(visualization_service); users has-not ?USER]
--->
?USER + ?SERVICE:[users has ?USER]
```

Questa reazione serve semplicemente a collegare a un servizio di visualizzazione un nuovo utente nel momento in cui questi si manifesta al sistema. Essendo il servizio finalizzato a recuperare le preferenze dell'utente l'eco-law non scatta qualora tale proprietà risulti vuota. L'id dell'utente viene così aggiunto alla proprietà `users` del servizio, il quale è così in grado di osservarla.

```
[BOND-RS] Bonds a new resource to a visualization service
?RESOURCE:[type=(resource); topic=?{TOP: ?TOP != ()}] +
?SERVICE:[type=(visualization_service); available_resources has-not ?RESOURCE]
--->
?RESOURCE + ?SERVICE:[available_resources has ?RESOURCE]
```

Similmente alla precedente, con questa eco-law, un servizio di visualizzazione ottiene un collegamento agli LSA di tipo risorsa presenti nel nodo. L'id di ognuno di essi viene inserito nella proprietà `available_resources`. Una volta ottenuti collegamenti a utenti e contenuti presenti sul nodo, il servizio può avviare la logica per la determinazione del contenuto da visualizzare basato sugli argomenti che mostrano maggiori preferenze. Quando tale decisione viene maturata il servizio imposta la proprietà `res_to_visualize` con il contenuto da mostrare e conseguentemente la seguente eco-law si attiva.

```
[VISUAL] Sets the content to visualize in a ready display
?DISPLAY:[type=(display); visualization=(ready); resource has-not ?RESOURCE] +
?SERVICE:[type=(visualization_service); res_to_visualize=?RESOURCE] + ?RESOURCE
--->
?DISPLAY:[visualization=(on); resource has ?RESOURCE] + ?SERVICE + ?RESOURCE
```

Questa eco-law imposta il contenuto scelto dal servizio di visualizzazione per essere mostrato su un display pronto, collegando l'LSA della risorsa da mostrare a una proprietà del display stesso. Il display che sarà pensato e realizzato come una entità reattiva, trovandosi il proprio LSA modificato, andrà a recuperare la risorsa e inizierà la visualizzazione.

Per quanto riguarda i contenuti informativi di carattere personale occorrerà un secondo set di eco-law che lavoreranno affiancando le prime per ottenere il risultato voluto.

```
[DET-US] Detects a user very close to the display and finds his flight number and inserts
         the informations of the flight in the display LSA

?USER:[type=(user); position=?POSITION; flight=(?NFL)] +
?POSITION:[type=(localization); signal=?{SIG: ?SIG > -40}] +
?DISPLAY:[type=(display); information has-not ?INFO] +
?FLIGHT:[type=(flight); flight_number=(?NFL); info=(?INFO)]
--->
?USER + ?ORIENTATION + ?DISPLAY:[information has ?INFO] + ?FLIGHT
```

L'eco-law rileva la vicinanza di un utente basandosi sulla potenza del segnale rilevata. Ciò è possibile confrontando il contenuto della proprietà `signal` dell'LSA di localizzazione dell'utente con un valore di soglia definito a priori. Se l'utente, oltre a trovarsi vicino al nodo/schermo, possiede anche una proprietà settata con il numero di un volo valido (ricordiamo di essere in un aeroporto) le informazioni relative a quest'ultimo vengono aggiunte allo schermo così che le possa mostrare (a parte rispetto allo stream primario governato dalle precedenti eco-law).

```
[GOAWAY-US] Detecst when a very close user move away and removes the his flight
            information from the display
?USER:[type=(user); position=?POSITION; flight=(?NFL)] +
?POSITION:[type=(localization); signal=?{SIG: ?SIG <= -40}] +
?DISPLAY:[type=(display); information has ?INFO]
?FLIGHT:[type=(flight); flight_number=(?NFL); info=(?INFO)]
--->
?USER + ?ORIENTATION + ?DISPLAY:[information has-not ?INFO] + ?FLIGHT
```

Dualmente alla precedente, l'eco-law sopra riportata rimuove le informazioni di un volo presente su un LSA utente quando questi si allontana dal display, azione rilevata dalla diminuzione del segnale nell'LSA `localization`.

5.2 Contenuto che segue l'utente

Il secondo caso di studio che si vuole presentare può essere visto come una estensione del caso precedente. In questo scenario abbiamo più display pubblici, che come visto precedentemente, adattano il loro contenuto basandosi sul profilo dell'utente e un unico utente che si muove tra essi. L'innovazione risiede nel fatto che il contenuto selezionato per l'utente da un display lo "segua" quando questi si avvicina al successivo e conseguentemente la visualizzazione su tale schermo riparta da dove era si era fermata su quello precedente. Per realizzare ciò, in seguito all'allontanamento da un display, occorre *diffondere* nel sistema l'informazione che l'*Utente U* ha interrotto la visione del *Contenuto C* all'istante di *Tempo T*, così che se la presenza dell'utente è rilevata nuovamente la visione possa riprendere.

5.2.1 Realizzazione dell'applicazione

La realizzazione di questa applicazione per certi versi risulta simile a quella del punto precedente per quanto riguarda l'adattamento del contenuto sulla base dell'utente. Vista la presenza di un solo utente si è pensato di semplificare il sistema prevedendo un collegamento diretto dei contenuti al display, evitando così di passare da un servizio apposito poiché in questo scenario non esiste la necessità di valutare il miglior contenuto da mostrare. Differentemente l'attenzione in questa applicazione si focalizza sulla realizzazione del meccanismo che permette la creazione e il passaggio delle informazioni relative alla "history" dell'utente. Tale obiettivo può essere raggiunto con due metodi differenti come esemplificato in figura 5.2:

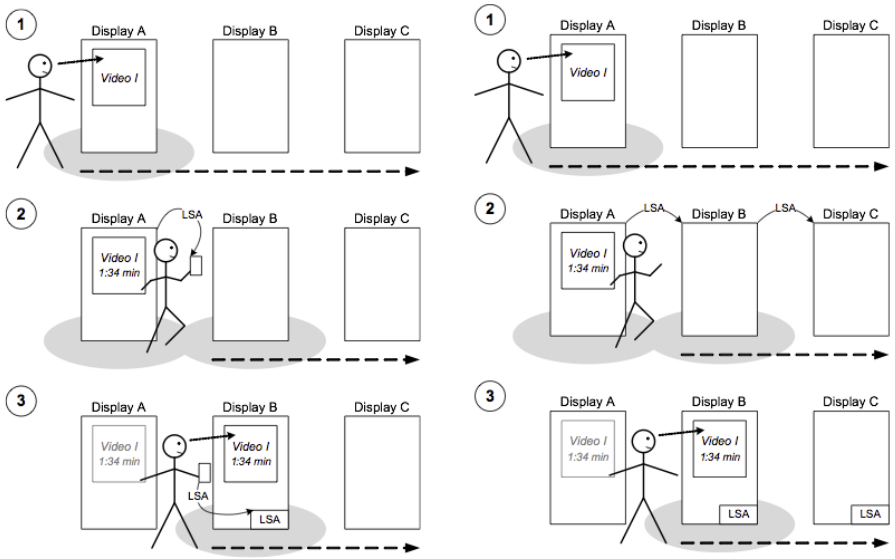

Figura 5.2: Tipologie del passaggio della history nel secondo caso di studio

- **la history è mantenuta sul device dell'utente**: in tal modo non occorre prevedere una comunicazione tra i nodi ma è l'utente stesso che allontanandosi "inconsapevolmente" porta con sé le informazioni relative a cosa già visionato. Il vantaggio di tale approccio è che l'informazione non viene diffusa in tutto il sistema senza appesantire il sistema con inutili diffusioni verso i display che non saranno raggiunti dall'utente. D'altro canto, la realizzazione di questo modello prevede la possibilità di accedere in scrittura al device dell'utente che in alcuni contesti può rappresentare una necessità non tollerata.

- **la history è diffusa da display a display**: in questo modo è il sistema a occuparsi della creazione della history e della sua diffusione tra i nodi per raggiungere tutti i display. Vantaggi e svantaggi sono duali al caso precedente. L'informazione verrà recapitata in tutti i nodi generando un numero elevato di diffusioni, per la maggior parte non utili, ma al proprio arrivo in qualsiasi nodo da parte dell'utente, la sua history sarà già presente.

L'approccio scelto per la realizzazione è stato il secondo. Tale motivazione è sorta anche dal fatto che per esigenze di tempo e di strumentazione a disposizione potendo disporre di due soli nodi e pertanto il numero di diffusioni generate in fase di creazione di un history sarebbe stato minimo. Vista la profonda somiglianza della maggior parte dei componenti con quelli presenti nell'esempio precedente, per brevità si riporta solo la struttura che avrà l'LSA che rappresenta lo storico dell'utente. Le interazione del sistema saranno poi spiegate commentando le eco-law utilizzate.

History

Questo LSA rappresenta lo storico di un utente del sistema dal punto di vista della visualizzazione di un contenuto. Questa informazione viene creata autonomamente quando si avverte che l'utente si sta allontanando e viene inviata ai nodi vicini tramite delle diffusioni. Le informazioni che contiene devono permettere di risalire non solo all'utente ma anche al contenuto che stava visionando e all'istante in cui la visione si è interrotta.

Le proprietà presenti nel LSA history sono:

- **type:** (`history`);
- **user:** contiene l'identificatore dell'utente a cui lo storico è riferito;
- **content:** contiene il nome/percorso della risorsa video che l'utente stava visionando;
- **time:** indica il tempo in secondi a cui si è fermata la visione nel display precedente.

5.2.2 Eco-law utilizzate

Di seguito si descriveranno brevemente le eco-law utilizzate nel caso di studio in esame per ottenere il comportamento atteso.

```
[VIS-R] Visualize a content on a display based on user preferences
?USER:[type=(user); preferences=?PREF; position=?POSITION}] +
?POSITION:[type=(localization); signal=?{SIG: ?SIG > -50}] +
?DISPLAY:[type=(display); visualization=(ready); content has-not ?CONT] +
?RESOURCE:[type=(resource); topic=?PREF; content=?CONT]
--->
?USER + ?POSITION + ?RESOURCE +
?DISPLAY:[visualization=(on); content has-not ?CONT; user=?USER; time=(0)]
```

Non dovendo utilizzare un service per la scelta di un contenuto da mostrare ciò viene realizzato con una singola eco-law la quale determina un contenuto che abbia un topic che fa match con le preferenze indicate dall'utente. L'utente deve essere inoltre abbastanza vicino al display, ciò viene rilevato utilizzando le proprietà dell'LSA `localization`. A differenza dell'esempio precedente non viene linkato al display l'intera LSA della risorsa da mostrare ma solo il contenuto che rappresenta il percorso tramite il quale potervi accedere. Con l'esecuzione di questa eco-law inizierà quindi l'esecuzione del video sul display.

```
[VIS-H] Visualize a content based on an history for a user
?USER:[type=(user)] + ?DISPLAY:[type=(display); visualization=(ready)] +
?RESOURCE:[type=(resource); content=?CONT] +
?HISTORY:[type=(history); time=?TIME; user=?USER; content=?CONT]
--->
?USER + ?RESOURCE +
?DISPLAY:[visualization=(on); content=?CONT; user=?USER; time=?TIME]
```

Questa eco-law prevede invece la visualizzazione di un contenuto associato a uno storico. L'LSA del display viene settato con le informazioni provenienti dalla history e questa viene cancellata poiché ormai ha compiuto il suo scopo nel sistema.

```
[PRE-MOVE] Stops the display when the users begin to move towards another display
?USER:[type=(user); position=?POSITION] +
?POSITION:[type=(localization); signal=?{SIG: ?SIG < -60}] +
?DISPLAY:[type=(display); visualization=(on); user=?USER] +
--->
?USER + ?POSITION + ?DISPLAY:[visualization=(idle)]
```

Quando un utente, che sta visualizzando un filmato, si allontana dal display, la diminuzione della potenza del segnale rilevato fa scattare l'eco-law settando lo stato del display su `idle`. Questo reagirà bloccando la visualizzazione e inserendo le informazioni sullo storico della visualizzazione nel proprio LSA per abilitare la seguente eco-law.

```
[HIST] Creates the history based on the display information
?DISPLAY:[type=(display); visualization=(paused); user=?USER; content=?CONT; time=?TIME] +
?NEIGH:[type=(#neighbour); where=?L]
--->
?DISPLAY:[visualization=(ready); user=(); content=(); time=(0.0)] + ?NEIGH
?HISTORY:[type=(history); #location=L1; time=?TIME; user=?USER; content=?CONT]
```

L'eco-law realizza lo storico basandosi sulle informazioni rese disponibili dal display e lo invio direttamente al nodo vicino (trattandosi di un applicazione che utilizza solo due nodi). Per fare ciò occorre che venga rilevata

l'LSA `neighbour` del nodo vicino così che la `location` dell'LSA da diffondere possa essere settata al giusto valore.

Il set di eco-law realizzato per l'applicazione è molto ristretto per motivo del ridotto numero di nodi utilizzati. Qualora se ne volessero gestire in numero maggiore occorrerebbe realizzare modificare le eco-law come segue:

- modificare HIST perché realizzi lo storico solo sul nodo corrente;
- introdurre una nuova eco-law che diffonde uno storico su un nodo a tutti i vicini;
- per evitare il flooding di copie dello stesso storico realizzare una eco-law che a parità degli altri dati elimina lo storico con il `time` minore.

5.3 Note finali

Il sistema con le poche eco-law riportate e commentate e con entità software molto semplici realizza entrambi i casi di studio voluti. Risulta così chiaro come il linguaggio delle Eco-law, grazie alla sua espressività, renda molto semplice la realizzazione di complesse interazioni tra componenti, anche semplici, permettendo la realizzazione di pattern di coordinazione non banali.

Capitolo 6

Conclusioni e sviluppi futuri

Rispetto alla realizzazione del prototipo iniziale, il modello e il linguaggio delle eco-law ha subito varie modifiche ed espansioni. Ciò ha richiesto una estesa fase di analisi per determinare quali aspetti necessitassero solo di una parziale revisione per aderire al nuovo modello, e quali occorreva riprogettare interamente. Il divario presente tra il nuovo modello computazionale e operazionale e quello a cui il prototipo faceva riferimento, ha obbligato ad una riscrittura completa della parte deputata alla rappresentazione ed esecuzione delle eco-law, con l'aggiunta di nuove entità e adattamenti di numerose altre. In particolare ciò è dovuto alla maggiore espressività offerta dal nuovo modello e all'aggiunta di funzionalità che hanno potenziato il linguaggio, con conseguente aumento di complessità soprattutto in riferimento ai meccanismi di determinazione del matching tra reagenti ed LSA e della creazione dei prodotti.

Il processo di realizzazione della meccanica delle eco-law è stato realizzato seguendo meticolosamente il modello computazionale astratto entità per entità, principalmente con una strategia bottom-up. Ciò ha spesso comportato l'introduzione di modifiche anche in entità esterne al modello delle eco-law, quali gli LSA e i loro contenuti. Ove possibile molti componenti esterni sono stati riutilizzati o riscritti lasciando quanto più immodificate le interfacce così da massimizzare il riuso degli altri componenti.

Il prototipo finale ottenuto risulta essere di buona qualità, supportando a pieno la realizzazione ed esecuzione delle eco-law secondo il modello descritto in [11]. Inizialmente, test sono stati condotti tramite simulazioni del sistema in locale, non disponendo della possibilità pratica di poterlo utiliz-

zare in distribuito per mancanza di tempo, mezzi e componenti software. La successiva integrazione di quanto realizzato in un prototipo dotato di moduli di comunicazione wireless e bluetooth ha permesso di testare il sistema realmente, così da mettere in luce limitazioni non riscontrabili in locale e poterle risolvere. Lo sviluppo di questa unione in unico progetto ha permesso di realizzare casi di studio non triviali, come mostrato nel capitolo 5.

Alla luce di tutto ciò il prototipo non è comunque privo di imperfezioni e possibili punti di intervento da realizzarsi in lavori futuri. In un sistema di questo tipo uno degli aspetti cruciali risiede, indubbiamente, nell'efficienza, che non è sempre massimizzata. In particolare, il componente *ReagentMatcher* risulta profondamente inefficiente nella parte che riguarda la gestione della coda degli eventi poiché ad ogni esecuzione corrisponde la sua reinizializzazione. Ciò potrebbe essere risolto prevedendo la realizzazione di grafi delle dipendenze così da determinare, a fronte di ogni modifica nel nodo, quanti e quali eventi già schedulati dovranno essere rimossi o modificati. Altro metodo per allegerire il carico computazionale potrebbe essere l'esecuzione del matching su più flussi separati di controllo a cui fornire subset delle eco-law da elaborare. La realizzazione di un componente avanzato pensato appositamente per realizzare il matching semantico è un altra necessità principale: aggiungendo un entità di questo tipo accompagnata da una opportuna modifica alla rappresentazione interna degli LSA si avrebbe un incremento sensibile dal punto di vista delle funzionalità e dell'espressività offerte dal linguaggio delle eco-law. In ultimo, anche la realizzazione di un interprete che comprenda la sintassi delle eco-law e le traduca nella corrispettiva rappresentazione Java sarebbe molto utile. Ciò permetterebbe a futuri utilizzatori di ignorare la struttura delle classi realizzata potendo indicare direttamente le eco-law desiderate nel linguaggio astratto. La realizzazione di tale interprete darebbe anche la possibilità di inserire dinamicamente, tramite interfacce grafiche per esempio, eco-law all'interno di nodi in esecuzione per modificarne il comportamento in tempo reale.

Elenco delle figure

Bibliografia

[1] Rdf primer. http://www.w3.org/TR/rdf-primer/.

[2] Sapere project. http://www.sapere-project.eu/.

[3] Bravetti, M., Gorrieri, R., Lucchi, R., and Zavattaro, G. Quantitative information in the tuple space coordination model. *Theoretical Computer Science 346*, 1 (2005), 28 – 57.

[4] Gillespie, D. T. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry 81*, 25 (1977), 2340–2361.

[5] Nardini, E., Viroli, M., Castelli, G., Mamei, M., and Zambonelli, F. A coordination approach to spatially-situated pervasive service ecosystem. In *The 13th International Conference COORDINATION'11* (Reykjavik, Iceland, 6-9 June 2011).

[6] Santarelli, M. Un'infrastruttura di coordinazione basata su regole chimico-semantiche. Master's thesis, Alma Mater Studiorum - Università di Bologna, Cesena, Italy, Mar. 2011.

[7] Stevenson, G., Dobson, S., Viroli, M., and Nardini, E. An approach based on web standards to the semantic coordination of pervasive service ecosystems. In *10th International Workshop on the Foundations of Coordination Languages and Software Architectures (FOCLASA 2011)*. Aachen, Germany, 10Sept. 2011.

[8] Tosi, A. Un'infrastruttura a spazi di tuple per ecosistemi di servizi pervasivi. Master's thesis, Alma Mater Studiorum - Università di Bologna, Cesena, Italy, Dec. 2010.

[9] Viroli, M., and Casadei, M. Biochemical tuple spaces for self-organising coordination. In *Coordination Languages and Models*, J. Field and V. T. Vasconcelos, Eds., vol. 5521 of *LNCS*. Springer, Lisbon, Portugal, 1June 2009, pp. 143–162. 11th International Conference (COORDINATION 2009), Lisbon, Portugal, June 2009.

[10] Viroli, M., Casadei, M., Nardini, E., and Omicini, A. Towards a chemical-inspired infrastructure for self-* pervasive applications. In *Self-Organizing Architectures*, D. Weyns, S. Malek, R. de Lemos, and J. Andersson, Eds. Springer, July 2010, ch. 8, pp. 152–176. 1st International Workshop on Self-Organizing Architectures (SOAR 2009), Cambridge, UK, 14-17 September.

[11] Viroli, M., Montagna, S., Pianini, D., and Stevenson, G. Deliverable d1.1, Oct. 2011.

[12] Viroli, M., Nardini, E., Castelli, G., Mamei, M., and Zambonelli, F. A coordination approach to adaptive pervasive service ecosystems. In *The 5th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2011)*. Ann Arbor, Michigan, USA, 3–8 Oct. 2011.

[13] Viroli, M., and Zambonelli, F. A biochemical approach to adaptive service ecosystems. *Information Sciences 180*, 10 (2010), 1876–1892.

[14] Viroli, M., Zambonelli, F., Casadei, M., and Montagna, S. A biochemical metaphor for developing eternally adaptive service ecosystems. In *24th Annual ACM Symposium on Applied Computing (SAC 2009)* (Honolulu, Hawai'i, USA, 8-12 Mar. 2009), S. Y. Shin, S. Ossowski, R. Menezes, and M. Viroli, Eds., vol. II, ACM, pp. 1221–1222.

[15] Weiser, M. Human-computer interaction. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995, ch. The computer for the 21st century, pp. 933–940.

Ringraziamenti

Un primo doveroso ringraziamento va al mio relatore prof. Mirko Viroli per la sua competenza, disponibilità e pazienza nei miei confronti durante il periodo in cui ho lavorato alla tesi.

Un grandissimo grazie anche ai miei correlatori Elena Nardini e Danilo Pianini per l'aiuto fornitomi. In particolare ringrazio Elena per la disponibilità che mi ha sempre dimostrato nel breve periodo in cui abbiamo lavorato fianco a fianco, nonostante i suoi mille impegni, e per il suo costante supporto morale. Ringrazio Danilo per la sua grande competenza e per la sua follia nonché per l'opera di correzione degli strafalcioni scritti nella versione iniziale della tesi.

Ringrazio i miei genitori per essermi sempre stati vicini, per avermi sostenuto, spronato, supportato e soprattutto sopportato in questo percorso della durata di ormai sette anni. Se arrivo a questo traguardo oggi in massima parte il merito è vostro, grazie!

Voglio poi ringraziare gli amici, di ieri e di oggi, quelli di San Bartolo, delle elementari, delle medie, dell'ITI, dell'Uni, di D&D, di WoW e tutti gli altri! Per la compagnia, per le bevute, per le serate, per le scemate fatte insieme, per le vacanze, per le nerdate, per esserci quando c'è bisogno vi ringrazio immensamente tutti! Paolo, Bondo, Santa, Cri, Giulia, Giacomo, Barbara, Benzo, Gatta, Zagno, Sarti, Giova, Davide, Ulex, Dappo, Piccia, Caro, Pagu.....vorrei nominarvi tutti ma siete davvero troppi!

Ringrazio il mio collega, Michele Morgagni, per l'impegno profuso nella tesi e in particolare nella realizzazione dei casi di studio, poiché se questa tesi ha un che di valido è anche merito suo.

Un ultimo ringraziamento va ai parenti, conoscenti e tutti coloro che vorrei citare, ma che sicuramente ho dimenticato, per il sostegno e gli incoraggiamenti che mi avete fornito in questi lunghi anni di carriera universitaria.

Matteo,
3 Ottobre 2011