Dynamic Distributed Nonmonotonic Multi-Context Systems*

Minh Dao-Tran Thomas Eiter Michael Fink
Thomas Krennwallner
Institut für Informationssysteme, Technische Universität Wien
Favoritenstraße 9-11, A-1040 Vienna, Austria
{dao,eiter,fink,tkren}@kr.tuwien.ac.at

Abstract

Nonmonotonic multi-context systems (MCS) provide a formalism to represent knowledge exchange between heterogeneous and possibly nonmonotonic knowledge bases (contexts). Recent advancements to evaluate MCS semantics (given in terms of so-called equilibria) enable their application to realistic and fully distributed scenarios of knowledge exchange. However, the current MCS formalism cannot handle open environments, i.e., when knowledge sources and their contents may change over time and are not known a priori. To improve on this aspect, we develop Dynamic Nonmonotonic Multi-Context Systems, which consist of schematic contexts that allow to leave part of the information interlinkage open at design time. A concrete interlinking is established by a configuration step at run time, where concrete contexts and information imports between them are fixed. We formally develop a corresponding extension and provide semantics by instantiation to ordinary MCS. Furthermore, we develop a basic distributed configuration algorithm and discuss several refinements that affect the resulting configurations, in particular by means of optimizations according to different quality criteria. This discussion is complemented with experimental results obtained with a corresponding prototype implementation.

1 Introduction

Developing modern knowledge-based information systems increasingly requires software engineers to handle knowledge integration tasks for accessing and aligning relevant information for particular application domains. As a result of recent developments of the World Wide Web, this information is in general distributed and comes with heterogeneous representation formalisms. Moreover, access to larger bodies of acquired knowledge is often organized via suitable interfaces, rather than providing direct access to the respective knowledge base.

*This research has been supported by the Austrian Science Fund (FWF) project P20841 and the Vienna Science and Technology Fund (WWTF) project ICT08-020.

Nonmonotonic multi-context systems (MCS) of Brewka and Eiter [2007] provide a formalism to address such knowledge integration tasks in a principled way. They generalize seminal work by Giunchiglia and Serafini [1994] and Roelofsen and Serafini [2005] that served the purpose to integrate different monotonic inference systems, into a heterogeneous MCS. Intuitively, individual knowledge bases—for historic reasons called *contexts*—are represented in a logic formalism with associated *belief sets* as a high-level representation of the associated semantics. By reference to such beliefs, so-called *bridge rules* allow to model an information flow between contexts. Semantics is given to an MCS in terms of *equilibria*, i.e., belief sets—one for each context—such that each belief set is acceptable for the respective local knowledge base and the information flow is in equilibrium.

The initial MCS approach has been gradually extended, in particular allowing for nonmonotonicity, both at the level of bridge rules as well as in knowledge bases of homogeneous contexts Roelofsen and Serafini [2005], Brewka, Roelofsen, and Serafini [2007], and more recently to accommodate heterogeneity of context logics also in the nonmonotonic case [Brewka and Eiter, 2007]. Furthermore, the practical importance of distributed settings has been perceived; they play e.g. an important role in ambient computing, where Antoniou, Papatheodorou, and Bikakis [2010] and Bikakis and Antoniou [2010] propose multi-context systems to integrate different entities, and develops a semantics for conflict resolution based on defeasible reasoning. In our work, the importance of distributed settings is reflected in the development of fully distributed evaluation algorithms for computing equilibria of nonmonotonic MCS [Dao-Tran, Eiter, Fink, and Krennwallner, 2010].

However, a characteristic that comes with many distributed application scenarios is that the environment is open, at least to some extent, meaning that participating knowledge sources and their contents may change over time and are not known a priori. This is in contrast with the static nature of current MCS in the sense that participating contexts and the corresponding information exchange need to be fixed completely at design time. Thus, atoms in bridge rules always point to a particular belief from a concrete context. This is prohibitive to formalizing systems where part of the behavior is instantiated at run-time only, as motivated by the following example.

Example 1 At the beginning of each semester, students in a group (including Alice, Bob, and Carol) need to choose courses from their curriculum. For each possible course, the students have three possible decisions, namely *select*, *hesitate*, and *eliminate* (in decreasing order). Intuitively, there is a potential for selecting a course if one finds it interesting. However, if the lecturer is known to be hard to please, they fear that it might be tough (or impossible) to get good marks and potentially eliminate the course. If there are reasons for both selecting and eliminating—or none—they are then in the state of hesitation, which dominates the other two potential decisions.

Moreover, the final decision of each student is supported by the decisions of their friends. If some friend gives a positive (resp., negative) opinion about a particular course, and no other friend shares an opposite opinion, then the group will adjust their final decision accordingly.

According to this strategy, the students do not specify in advance for a course which friends they will consult. This depends on the friends they will meet at the course orientation meeting. While attending the orientation meeting and exchanging opinions, every student in the group finally comes up with a list of courses that

conforms with the choices of their colleagues.

For example, Alice may believe that if Bob hesitates or selects a course, then this is a positive sign, because he is very cautious; on the other hand, if Bob eliminates a course, then this is a negative sign. But she has a different opinion about Carol's choice, namely she is encouraged only when Carol selects the course and is discouraged otherwise. Carol, who is a bit more careful, might only accept that Bob's selection (resp., elimination) of the course as a positive (resp., negative) hint, i.e., she has no bias when Bob is hesitating. Finally, Bob may interpret the opinions of the two girls in the same way as Carol does with his, i.e., mapping selections to be positive, elimination to be negative, and having no preference w.r.t. hesitance.

When the three of them talk about the course on Answer Set Programming, which Bob finds interesting, but Alice has the impression that the professor is very demanding, they ask Carol, who has no additional opinion about it. One of the outcomes of the discussion is that Bob and Carol will select the course while Alice hesitates.

The current MCS setting is sufficient to formalize the last part of the discussion between Alice, Bob, and Carol (see also Example 3), but lacks dynamicity to formalize the general setting of Example 1.

In this work, we address the above shortcoming of the MCS formalism concerning open environments of information exchange, that is when at design time the concrete knowledge sources participating in an information exchange are not known. Intuitively, what is needed to cope with such scenarios is a formalism for information exchange which is closer towards a peer-to-peer (P2P) approach, where so-called peers can at any time join or leave the system dynamically [Aberer, Punceva, Hauswirth, and Schmidt, 2002]. To this end, we present *Dynamic Nonmonotonic Multi-Context Systems*, which consist of schematic contexts that may leave some information interlinkage open at design time; this linkage is established by a configuration step at run time, in which concrete contexts and information imports between them are wired.

More specifically, our contributions are the following:

- We formalize dynamic multi-context systems, which extend the MCS formalism with so-called *schematic bridge rules*. Intuitively, schematic bridge rules may contain place holders that can range over both context identifiers and beliefs. Their semantics is defined via suitable notions of substitution and binding, where a *context substitution* maps context holders to concrete contexts and a binding maps schematic belief atoms to adequate concrete beliefs. To take into account that a perfectly matching belief might not exist, we use (unless exact substitution is forced) a 'similar'-based binding beliefs in which schematic beliefs are bound to 'similar' beliefs, which is assessed by a similarity function. To determine such beliefs, we foresee a matchmaking component as an oracle which returns on a call a list with similar beliefs. More precisely, it provides simple term substitutions according to an underlying similarity measure.

- We consider the problem of finding an instantiation of a dynamic multi-context system, starting from a specific context, i.e., a concrete "configuration" of the (open) system. To solve it, we first present a basic algorithm for computing configurations of dynamic MCS. As the number of configurations can be very large in general, we then consider different heuristics to generate 'good' ones,

which take topological structure and/or different criteria of qualities of individual matches (bindings) into account. The algorithm is fully distributed, i.e., instances run at different contexts, and the configurations are found by local computations plus communication.

- Finally, we present some results of an experimental prototype implementation of the configuration algorithm under different heuristics for the configuration. The results show that the latter behave on a few considered topologies of potential interconnections as expected, and may also lead in particular cases to configurations which corresponds to natural social information interlinkage.

Using dynamic MCS, a broader range of application scenarios can be modeled which require the flexibility of taking changing context into account. In particular, group formation to satisfy information needs of heterogeneous components, with possible selection among different alternatives, can be readily expressed.

The remainder of this paper is structured as follows. The next section recalls basic concepts of answer set programming (ASP, the context logic of choice in our examples) and nonmonotonic MCS which are the starting point of this work. Dynamic MCS are subsequently introduced in Section 3, where we provide their formal definition and semantics in terms of instantiation to ordinary MCS. Section 4 contains then the description of our basic distributed configuration algorithm and discussions of refinements such as, e.g., different heuristics to drive the configuration. Furthermore, in Section 5 we report on a prototype implementation of the algorithm and some experimental results. Conclusions and issues for further work are eventually given in Section 6.

2 Preliminaries

We recall some basic notions of disjunctive logic programs under the answer set semantics [Gelfond and Lifschitz, 1991] and heterogeneous nonmonotonic multi-context systems [Brewka and Eiter, 2007].

Answer Set Programs. Let $\mathcal{A}$ be a finite alphabet of atomic propositions. A *disjunctive rule* r is of the form

$$a_1 \vee \cdots \vee a_k \leftarrow b_1, \ldots, b_m, \text{not } b_{m+1}, \ldots, \text{not } b_n \ , \tag{1}$$

$k + n > 0$, where all a_i and b_j are atoms from $\mathcal{A}$.[1] We let $H(r) = \{a_1, \ldots, a_k\}$, and $B(r) = B^+(r) \cup B^-(r)$, where $B^+(r) = \{b_1, \ldots, b_m\}$ and $B^-(r) = \{b_{m+1}, \ldots, b_n\}$. An *answer set program* P is a finite set of rules r of form (1).

An *interpretation* for P is any subset $I \subseteq \mathcal{A}$. It *satisfies* a rule r, if $H(r) \cap I \neq \emptyset$ whenever $B^+(r) \subseteq I$ and $B^-(r) \cap I = \emptyset$. I is a *model* of P, if it satisfies each $r \in P$.

The *GL-reduct* [Gelfond and Lifschitz, 1991] P^I of P relative to I is the program obtained from P by deleting (i) every rule $r \in P$ such that $B^-(r) \cap I \neq \emptyset$, and (ii) all $\text{not } b_j$, where $b_j \in B^-(r)$, from every remaining rule r.

[1] Gelfond and Lifschitz [1991] used classical literals as basic constituents rather than atoms. For simplicity, we disregard classical (also called strong) negation here; this does not affect the expressiveness of the formalism.

An interpretation I of a program P is called an *answer set* of P iff I is a $\subseteq$-minimal model of P^I.

Example 2 We will now model Example 1 as an answer set program. Let R_i be a set of the following rules:

$$
\begin{array}{ll}
s_i \leftarrow ps_i, \text{not } h_i, \text{not } e_i & e_i \leftarrow pe_i, \text{not } h_i, \text{not } inc_i \\
s_i \leftarrow ph_i, \text{not } ps_i, \text{not } h_i, \text{not } e_i, inc_i & e_i \leftarrow ph_i, dec_i \\
h_i \leftarrow ps_i, \text{not } e_i, dec_i & ps_i \leftarrow inter_i \\
h_i \leftarrow ph_i, \text{not } inc_i, \text{not } dec_i & pe_i \leftarrow hprof_i \\
h_i \leftarrow pe_i, \text{not } ph_i, inc_i & ph_i \leftarrow ps_i, pe_i \\
 & ph_i \leftarrow \text{not } ps_i, \text{not } pe_i
\end{array}
$$

The atoms have the following meaning: s_i, h_i, and e_i stand for the three decisions: select, hesitate, and eliminate, resp. Similarly, ps_i, ph_i, and pe_i stand for the potential to select, hesitate, and eliminate a course, resp. A course is interesting if $inter_i$ is true, and a professor is hard to please if $hprof_i$ is true. The atoms inc_i and dec_i mean that a student inclines and declines to select a course, resp.

The program $P_1 = R_1 \cup \{hprof_1\}$ has one answer set $\{e_1, pe_1, hprof_1\}$, $P_2 = R_2 \cup \{inter_2\}$ has the answer set $\{s_2, ps_2, inter_2\}$, while $P_3 = R_3$ has the answer set $\{h_3, ph_3\}$.

Intuitively, P_1 represents Alice's mind. She thinks that the professor is hard to please ($hprof_1$), hence she potentially eliminates (pe_1) the course and will eliminate it (e_1) if no more support information is provided. On the other hand, P_2 represents Bob's mind. He is really interested in the course ($inter_2$) and selects it (s_2) based on his potential of selecting the course (ps_2). Carol, modeled by P_3, adds no personal view about the course. She is currently hesitating (h_3, ph_3) in taking the course; her final decision can change depending on decisions of other friends.

Multi-Context Systems. A *logic* is, viewed abstractly, a tuple $L = (\mathbf{KB}_L, \mathbf{BS}_L, \mathbf{ACC}_L)$, where

- $\mathbf{KB}_L$ is a set of well-formed knowledge bases, each being a set (of formulas),
- $\mathbf{BS}_L$ is a set of possible belief sets, each being a set (of formulas), and
- $\mathbf{ACC}_L \colon \mathbf{KB}_L \rightarrow 2^{\mathbf{BS}_L}$ assigns each $kb \in \mathbf{KB}_L$ a set of acceptable belief sets.

This covers many (non-)monotonic KR formalisms like description logics, default logic, answer set programs, etc.

For example, a (propositional) *ASP logic* L may be such that $\mathbf{KB}_L$ is the set of answer set programs over a (propositional) alphabet $\mathcal{A}$, $\mathbf{BS}_L = 2^{\mathcal{A}}$ contains all subsets of atoms, and $\mathbf{ACC}_L$ assigns each $kb \in \mathbf{KB}_L$ the set of all its answer sets.

Definition 1 *A* multi-context system (MCS) *is a set $M = (C_1, \ldots, C_n)$, consisting of* contexts *$C_i = (L_i, kb_i, br_i)$, such that $1 \leq i \leq n$, $L_i = (\mathbf{KB}_i, \mathbf{BS}_i, \mathbf{ACC}_i)$ is a logic, $kb_i \in \mathbf{KB}_i$ is a knowledge base, and br_i is a set of L_i-*bridge rules *of the form*

$$s \leftarrow (c_1 : p_1), \ldots, (c_j : p_j), \text{not } (c_{j+1} : p_{j+1}), \ldots, \text{not } (c_m : p_m) \qquad (2)$$

where $1 \leq c_k \leq n$ and p_k is an element of some belief set of L_{c_k} (i.e., $p_k \in \bigcup \mathbf{BS}_{L_{c_k}}$), $1 \leq k \leq m$, and $kb \cup \{s\} \in \mathbf{KB}_i$ for each $kb \in \mathbf{KB}_i$.

Informally, bridge rules allow to modify the knowledge base by adding s, depending on the beliefs in other contexts.

The semantics of an MCS M is defined in terms of particular *belief states*, which are sequences $S = (S_1, \ldots, S_n)$ of belief sets $S_i \in \mathbf{BS}_i$. Intuitively, S_i should be a belief set of the knowledge base kb_i; however, also the bridge rules br_i must be respected. To this end, kb_i is augmented with the conclusions of all $r \in br_{c_i}$ that are applicable.

Formally, r of form (2) is *applicable in* S, if $p_i \in S_i$, for $1 \leq i \leq j$, and $p_k \notin S_k$, for $j+1 \leq k \leq m$. Let $app(R, S)$ denote the set of all bridge rules $r \in R$ that are applicable in S, and $head(r)$ the part s of any r of form (2).

Definition 2 *A belief state $S = (S_1, \ldots, S_n)$ of a multi-context system M is an* equilibrium *iff for all $1 \leq i \leq n$, $S_i \in \mathbf{ACC}_i(kb_i \cup \{head(r) \mid r \in app(br_i, S)\})$.*

Example 3 Let $M' = (C_1, C_2, C_3)$ be an MCS such that all L_i are ASP logics, with alphabets $\mathcal{A}_i = \{s_i, h_i, e_i, ps_i, ph_i, pe_i, inter_i, hprof_i, inc_i, dec_i\}$. Suppose $kb_i = P_i$, with P_i taken from Example 2, and

$$br_1 = \left\{ \begin{array}{lcl} inc_1 & \leftarrow & (2 : s_2), \text{not } (3 : h_3), \text{not } (3 : e_3), \text{not } (1 : dec_1) \\ inc_1 & \leftarrow & (2 : h_2), \text{not } (3 : h_3), \text{not } (3 : e_3), \text{not } (1 : dec_1) \\ dec_1 & \leftarrow & (2 : e_2), \text{not } (3 : s_3), \text{not } (1 : inc_1) \end{array} \right\},$$

$$br_2 = \left\{ \begin{array}{lcl} inc_2 & \leftarrow & (1 : s_1), \text{not } (3 : e_3), \text{not } (2 : dec_2) \\ dec_2 & \leftarrow & (3 : e_3), \text{not } (1 : s_1), \text{not } (2 : inc_2) \end{array} \right\}, \text{ and}$$

$$br_3 = \left\{ \begin{array}{lcl} inc_3 & \leftarrow & (2 : s_2), \text{not } (1 : e_1), \text{not } (3 : dec_3) \\ dec_3 & \leftarrow & (1 : e_1), \text{not } (2 : s_2), \text{not } (3 : inc_3) \end{array} \right\}$$

One can check that $S = (\{h_1, pe_1, hprof_1, inc_1\}, \{s_2, ps_2, inter_2\}, \{s_3, ph_3, inc_3\})$ is an equilibrium of M'. Intuitively, M' models the discussion between Alice (C_1), Bob (C_2), and Carol (C_3). Comparing this to Example 2, the decision of Bob influences those of Alice and Carol, as Alice now hesitates about the course even though having the potential of eliminating it, while Carol decided to select the course although she was hesitating about it before.

3 Dynamic Nonmonotonic MCS

In the following section, we will develop a framework that caters for dynamics in Multi-Context Systems using placeholders for contexts and beliefs in schematic bridge rules. The open parts of such rules can be made concrete by linking them to ordinary MCS.

3.1 Basic Notions

Let $\mathcal{V}_{ctx}$ be a vocabulary of *context holders*,[2] and let $\Sigma = \bigcup \Sigma_i$ a set of (possibly shared) *signatures*. Unless stated otherwise, elements from $\mathcal{V}_{ctx}$ (resp., Σ) are denoted with first letter in upper case (resp., lower case). Furthermore, we define the set

[2]We use the term 'holder' rather than 'variable' to avoid confusion with variables as introduced for *relational MCS* [Michael Fink and Weinzierl, 2011].

$\Sigma_{@}$ (resp., $\Sigma_{\sim}$) of exact (resp., similar) *schematic beliefs* as the set of symbols $@[p]$ (resp., $[p]$) for all p in Σ. Let $bel(@[p]) = bel([p]) = p$ be a function for extracting the belief symbol from a schematic belief.

Definition 3 *A* dynamic multi-context system $M = \{C_1, \ldots, C_n\}$ *is a set of* schematic contexts $C_i = (L_i, kb_i, sbr_i)$*, where*

- $L_i = (\mathbf{KB}_i, \mathbf{BS}_i, \mathbf{ACC}_i)$ *is a logic based on a signature* Σ_i*,*
- $kb_i \in \mathbf{KB}_i$ *is a knowledge base, and*
- sbr_i *is a set of* L_i schematic-bridge rules *(*s-bridge rules *for short) of the form*

$$s \leftarrow B(r), \chi(r) \tag{3}$$

with $B(r) = (X_1 : P_1), \ldots, (X_j : P_j), \mathrm{not}\,(X_{j+1} : P_{j+1}), \ldots, \mathrm{not}\,(X_m : P_m)$*, where each* $sb_\ell = (X_\ell, P_\ell)$*,* $1 \leq \ell \leq m$*, is a* schematic bridge atom *(*s-bridge atom *for short) in which* $X_\ell \in M \cup \mathcal{V}_{ctx}$ *either refers to a context in* M *or is a context holder, and* $P_\ell \in \Sigma \cup \Sigma_{@} \cup \Sigma_{\sim}$ *is either a belief (i.e.,* $P_\ell \in \Sigma$*) or a schematic belief (*$P_\ell \in \Sigma_{@} \cup \Sigma_{\sim}$*); and* $\chi(r) = Y_{1_1} \neq Y_{1_2}, \ldots Y_{k_1} \neq Y_{k_2}$ *is a (possibly empty) list of inequality atoms* $Y_{i_1} \neq Y_{i_2}$ *(*$1 \leq i \leq k$*) where* Y_{i_1}, Y_{i_2} *are two different context holders from* $X_1, \ldots, X_m$*.*

For simplicity, we assume that context holders in rules are standardized apart, i.e., there exist no two context holders with the same name in two different rules, as they can be bound to different contexts.

Example 4 A group of n students in Example 1 can be modeled as a dynamic MCS $M = \{C_1, \ldots, C_n\}$, where, for each $C_i = (L_i, kb_i, sbr_i) \in M$, $kb_i = R \cup F_i$, with R from Example 2 and $F_i \subseteq \{inter, hprof\}$, and the following set of schematic bridge rules

$$sbr_i = \left\{ \begin{array}{lcl} inc_i & \leftarrow & (X_i : [pos_i]), \mathrm{not}\,(Y_i : [neg_i]), \mathrm{not}\,(i : dec_i), X_i \neq Y_i \\ dec_i & \leftarrow & (Z_i : [neg_i]), \mathrm{not}\,(T_i : [pos_i]), \mathrm{not}\,(i : inc_i), Z_i \neq T_i \end{array} \right\}.$$

The first rule expresses that student i should be inclined to take a course, if some student in the group has a positive opinion and some student does not have a negative opinion, and student i herself is not declining to take the course. The second rule is similar, but for declining the course.

Here, the context holders set is $\mathcal{V}_{ctx} = \{X_i, Y_i, Z_i, T_i\}$, the local signature at each context C_i is $\Sigma_i = \mathcal{A}_i \cup \{pos_i, neg_i\}$ with $\mathcal{A}_i$ taken from Example 3. We use here only similar schematic beliefs, namely $[pos_i]$ and $[neg_i]$.

Dynamic MCS differ from original MCS in the sense that s-bridge atoms in general are not specifically bound to some beliefs of other dynamic contexts in the system, but rather represent a collection of possibilities to point to different beliefs in other contexts. From a topological point of view, such a high-level representation incurs numerous dependencies between dynamic contexts in general. However, most of these dependencies are not reflected in intended instantiations, which provides evidencen not to aim at defining equilibria of dynamic MCS in a direct way. Hence, for defining semantics one rather considers how to bind them to original MCS. We consider such

bindings next, starting with the notion of binding a schematic bridge atom to ordinary bridge atoms based on potential matches.

A *context substitution* is a map $\sigma\colon (M \cup \mathcal{V}_{ctx}) \to M$ such that for every inequality atom $Y_{i_1} \neq Y_{i_2}$ occurring in bridge rules of a context $C \in M$, $\sigma(Y_{i_1}) \neq \sigma(Y_{i_2})$. For a context C_k, we denote by $\sigma|_{C_k}$ the *restriction* of σ to C_k, i.e., the subset of σ containing only maps from a context holder appearing in an s-bridge rule in C_k. Due to the assumption of standardization of context holders, the set of restrictions of σ to all individual contexts in M is a partitioning of σ.

The application of a context substitution σ to an s-bridge atom $sb = (X : P)$ is $\sigma(sb) = (C_j : P)$ where $P \in \Sigma_@ \cup \Sigma_\sim \cup \Sigma_j$, and either $X = C_j$ or $X \in \mathcal{V}_{ctx}$ satisfying $(X \mapsto C_j) \in \sigma$. Intuitively, the application of a context substitution is responsible for instantiating a potential context holder of an s-bridge atom.

Example 5 Let $\sigma = \{X_1 \mapsto C_2, Y_1 \mapsto C_3\}$ be a context substitution, then the applications of σ to $sb_1 = (X_1 : [pos_1])$ and $sb_2 = (Y_1 : [neg_1])$ are $sb'_1 = \sigma(sb_1) = (C_2 : [pos_1])$ and $sb'_2 = \sigma(sb_2) = (C_3 : [neg_1])$.

Let $f_M\colon \Sigma \times \Sigma \to [0, 1]$ be a function measuring the similarity between beliefs in an MCS M, where higher similarity of beliefs p and q is reflected by a larger value of $f_M(p, q)$. In particular, $f_M(p, q) = 1$ means that p and q are considered to have highest similarity (especially, if they are identical) and $f_M(p, q) = 0$ that p and q are completely dissimilar. We do not commit to a particular function f_M here, which may depend on the application; in what follows, we just assume that some such function f_M has been fixed and is available, for instance consider similarity of terms as defined by WordNet [Miller, 1995] or different types of matches on Larks [Sycara, Widoff, Klusch, and Lu, 2002] specifications.

Definition 4 *Given an MCS M, a similarity function f_M, and a threshold t, a* term substitution *from C_i to C_j in M w.r.t. f_M and t, denoted by $\eta^t_M(C_i, C_j)$, is a relation* $\eta^t_M(C_i, C_j) = \{(a, b) \mid a \in \Sigma_i, b \in \Sigma_j, f_M(a, b) > t\}$.

By η^t_M we denote the collection of all pairwise term substitutions in M. The *density* of M w.r.t. η^t_M is $d_{\eta^t_M} = |\{(C_i, C_j) \mid \eta^t_M(C_i, C_j) \neq \emptyset\}|$. In the sequel, we pick a default value $t = 0$; furthermore, we use η instead of η^0_M when M is clear from the context.

The application of a term substitution η to an s-bridge atom $sb = (C_j : P)$ in context C_i, denoted by $\eta(sb)$, is defined by (i) if $P = a$, then $\eta(sb) = \{(C_j : a)\}$ if $a \in \Sigma_j$, and $\emptyset$ otherwise; (ii) if $P \in \Sigma_@$, then $\eta(sb) = \{(C_j : b) \mid (bel(P), b) \in \eta(C_i, C_j), f_M(bel(P), b) = 1\}$; (ii) if $P \in \Sigma_\sim$, then $\eta(sb) = \{(C_j : b) \mid (bel(P), b) \in \eta(C_i, C_j)\}$. Intuitively, the application of a term substitution to an s-bridge atom only applies to s-bridge atoms with instantiated context holders, and then collects all possible substitutions for the schematic belief P.

Example 6 Continue with Example 5, suppose that we have a similarity function f_M whose interesting part is described in Table 1; and for the rest, f_M takes value 1 if the two parameters are identical and 0 otherwise.

The values of f_M are taken in conformity with the scenario in Example 1, e.g., Alice trusts the *select* and *hesitate* decisions of Bob as a positive sign at a measurement of 0.9 and 0.6, respectively. She considers Bob *eliminating* the course as a negative

Table 1: Interesting part of similarity function

f_M	s_1	h_1	e_1	s_2	h_2	e_2	s_3	h_3	e_3
pos_1	0.0	0.0	0.0	**0.9**	**0.6**	0.0	**0.7**	0.0	0.0
neg_1	0.0	0.0	0.0	0.0	0.0	**0.8**	0.0	**0.5**	**0.7**
pos_2	**0.7**	0.0	0.0	0.0	0.0	0.0	**0.6**	0.0	0.0
neg_2	0.0	0.0	**0.7**	0.0	0.0	0.0	0.0	0.0	**0.6**
pos_3	**0.6**	0.0	0.0	**0.8**	0.0	0.0	0.0	0.0	0.0
neg_3	0.0	0.0	**0.7**	0.0	0.0	**0.8**	0.0	0.0	0.0

sign of 0.8. Hence, $f_M(pos_1, s_2) = 0.9$, $f_M(pos_1, h_2) = 0.6$, and $f_M(neg_1, e_2) = 0.8$. On the other hand, Alice is encouraged only when Carol selects the course, but with less confidence as $f_M(pos_1, s_3) = 0.7$; and she interprets other choices from Carol as discouragement with $f_M(neg_1, h_3) = 0.5$, and $f_M(neg_1, e_3) = 0.7$. The next rows in Table 1 show the opinions of Bob and Carol about the decisions of the others. Note that they do not take *hesitance* into account.

The term substitutions from C_1 to C_2 and C_3 w.r.t. f_M are $\eta(C_1, C_2) = \{(pos_1, s_2), (pos_1, h_2), (neg_1, e_2)\}$, and $\eta(C_1, C_3) = \{(pos_1, s_3), (neg_1, h_3), (neg_1, e_3)\}$, respectively. The applications of these substitution to sb'_1 and sb'_2 are $\eta(sb'_1) = \{(C_2 : s_2), (C_2 : h_2)\}$ and $\eta(sb'_2) = \{(C_3 : e_3)\}$.

Based on σ and η, the notion of a bridge substitution is simply defined by their composition.

Definition 5 *Let σ be a context substitution of M. The* bridge substitution θ *for an s-bridge atom sb w.r.t. σ is $\theta(sb) = \eta(\sigma(sb))$.*

Thus, intuitively, the bridge substitution of an s-bridge atom is done in two steps. First, one uses σ to instantiate the context holders, and then η takes effect to instantiate the schematic beliefs.

Example 7 The bridge substitution of the s-bridge atom sb_1 from Example 5 w.r.t. σ from the same example and η from Example 6 is $\theta(sb_1) = \eta(\sigma(sb_1)) = \{(C_2 : s_2), (C_2 : h_2)\}$. Similarly, we have $\theta(sb_2) = \{(C_3 : e_3)\}$.

Let us now turn to bridge rules. Given a schematic bridge rule r of form (3) in a context C_i, a context substitution σ is called a substitution of r iff there exist bridge substitutions θ_ℓ w.r.t. σ for all schematic bridge atoms sb_ℓ in $B(r)$, i.e., for $1 \leq \ell \leq m$, such that $\theta_\ell(sb_\ell) \neq \emptyset$. The bindings of r w.r.t. σ are defined as the set of bound rules $r\sigma$ where each bound rule is obtained by replacing $sb_\ell = (X : P)$ in $B(r)$ with

(i) some bridge atom $(C_i : b)$ such that $(C_i : b) \in \theta_\ell(sb_\ell)$, if $sb_\ell \in B^+(r)$; and

(ii) the sequence of negated bridge atoms $\text{not}\,(C_i : b_1), \ldots, \text{not}\,(C_i : b_k)$ such that $\{(C_i : b_1), \ldots, (C_i : b_k)\} = \theta_\ell(sb_\ell)$; if $sb_\ell \in B^-(r)$.

The size m of $r\sigma$ is determined by $m = \Pi_{sb_\ell \in B^+(r)} |\theta_\ell(sb_\ell)|$.

Example 8 Continuing our example, pick r as the first s-bridge rule from Example 4 and consider it in context C_1 representing Alice's mind. Furthermore, regard the context substitution σ from Example 5. Taking the term substitutions of Example 6 into account, $r\sigma$ consists of two bound rules, namely the first two rules from br_1 in Example 3.

Given a set of s-bridge rules R of a context C and a context substitution σ, the binding of R w.r.t. σ is defined as $R\sigma = \bigcup_{r \in R} r\sigma$. Then, the binding of a context C w.r.t. a context substitution σ is given by $C\sigma = (kb, sbr\sigma)$.

Definition 6 *Given a dynamic MCS M and a context substitution σ, the set $M\sigma = \{C_1\sigma, \ldots, C_\ell\sigma\}$, is a* binding *of M w.r.t. a context C_k iff*

1. $C_k \in \{C_1, \ldots, C_\ell\}$
2. $\{C_1, \ldots, C_\ell\} \subseteq M$
3. *σ is a substitution for all s-bridge rules in all contexts $C_1, \ldots, C_\ell$, and*
4. $\{C_1, \ldots, C_\ell\} = \bigcup_{C_j \in \{C_1 \ldots C_\ell\}} \{C \mid r \in sbr_j\sigma \wedge (C : a) \in B(r)\} \cup \{C_k\}$.

A *belief state* of $M\sigma$ is a sequence of belief sets $S = (S_1, \ldots, S_\ell)$, one S_i for each $C_i\sigma$. Such a belief state is an *equilibrium* of M w.r.t. C_k and σ iff for all $1 \leq i \leq \ell$, it holds that $S_i \in \mathbf{ACC}_i(kb_i \cup head(r) \mid r \in app(sbr_i\sigma, S))$.

The quality of a binding is

$$\frac{1}{|\mathcal{U}|} \cdot \sum_{(a,b) \in \mathcal{U}} f_M(a, b)$$

where $\mathcal{U}$ is the set of matches used in the binding, i.e., $\mathcal{U} = \{(a, b) \mid a = bel(P) \wedge sb = (X : P) \in C_i, 1 \leq i \leq \ell \wedge (C_j : b) \in \theta(sb), 1 \leq j \leq \ell\}$.

Intuitively, a binding of M w.r.t. a substitution σ and a context C_k consists of a subset of the contexts of M, which must contain C_k (hence conditions 1 and 2), which is properly instantiated by θ (condition 3) and, moreover, *closed* in the sense that every selected context, except for C_k, is used for instantiating bridge rules of other chosen contexts (condition 4). The notions of belief state and equilibrium are then inherited from ordinary MCS. The quality of a binding is simply the average of the similarities of all matches used in the binding.

3.2 From Dynamic to Ordinary Multi-Context Systems

Recapturing the idea of binding a dynamic MCS M to an original one, starting from a context C_{root}, one needs

1. to know all potential neighbors C_j for a context C_i and the term substitutions $\eta(C_i, C_j)$ between them;
2. a strategy to start from C_{root} and to expand the system by: first determining a context substitution σ for each context term in the s-bridge rules of C_{root}, and then to continue the process at each neighbor, until a closed system is obtained;

3. some decision criteria to guide the process to come up with a most suitable substitution to bind M.

Task (1) is in fact matching beliefs from different contexts. This problem shares similarities with the *matchmaking problem* in Multi-Agent Systems (MAS), which has been widely considered [Sycara et al., 2002, Ogston and Vassiliadis, 2001]. Our work in this paper is not doing matchmaking but rather using the matchmaker as a building block to configure the inter-linkage between contexts in a dynamic MCS to form ordinary ones. As such, we assume that there exists a *matchmaker* MatchMaker which, upon a call MatchMaker(P, C_i) from a context C_i, returns a set of potential neighbors such that

- if P is a schematic variable in $\Sigma_@$, then N is the set of context names C_j where the term substitution $\eta(C_i, C_j)$ contains at least one pair $(bel(P), a)$ with $f_M(bel(P), a) = 1$;
- if P is a schematic variable in $\Sigma_\sim$, then N is the set of context names C_j where the term substitution $\eta(C_i, C_j)$ is nonempty;
- if $P = p$ is an atom from Σ, then N is the set of all contexts C_j such that $p \in \Sigma_j$.

Further queries to the matchmaker such as MatchMaker(C_i, C_j) can give back $\eta(C_i, C_j)$ and/or the value of f_M for the pairs of atoms from this term substitution. This information is used for calculating the quality of the system after instantiating.

The main problems that we solve in this paper are those in (2) and (3). Concerning (2), we present a backtracking algorithm to enumerate all possible context substitutions σ, in a distributed, peer-to-peer like setting. This means that each context, knowing only its potential neighbors by asking the matchmaker, can only locally choose the matches for its own s-bridge atoms, which consequently decides its real neighbors in the resulting MCS, and then has to ask these neighbors to continue the configuration (hence our algorithm is called lconfig).

The process starts at C_{root} and continues in a Depth-First Search (DFS) manner, carrying along the context substitution σ built up so far, until for all chosen contexts their s-bridge atoms are bound.

Regarding (3), we propose general methods to compare the outcome of different substitutions on two main aspects, namely (Q1) the matching quality of the bound rules and (Q2) the topological quality of the resulting MCS. These methods can be seen as heuristics that can be plugged into the basic version of lconfig to get the context substitutions returned ordered by quality.

For clarity and simplicity, in the sequel, we first present the very basic version of lconfig with generic possibilities for optimization. We then briefly go through such possibilities, where we choose some interesting ones to discuss in more detail and suggest potential realizations of them.

4 Multi-Context System Configuration

The question is now how one can actually compute substitutions as sketched above. We present a basic configuration algorithm which computes concrete bindings for a

Algorithm 1: lconfig(C_{root}, R, σ) at C_k

Input: C_{root}: root context, R: set of s-bridge rules, σ: context substitution
Output: context susbstitution for C_k
Data: $obuf_r$ for every $r \in R$: substitutions for r
if $R = \emptyset$ **then**
(a) $\quad \mathcal{C}_{new} := \mathsf{get\_contexts}(\sigma|_{C_k}) \setminus (\mathsf{get\_contexts}(\sigma \setminus \sigma|_{C_k}) \cup \{C_{root}\})$
$\quad$ **if** $\mathcal{C}_{new} \neq \emptyset$ **then return** invoke_neighbors$(C_{root}, \mathcal{C}_{new}, \sigma)$
$\quad$ **else return** $\{\sigma\}$
(b) **else**
(c) $\quad$ pick r from R, and $obuf_r := \mathsf{bind\_rule}(\chi(r), B(r), \sigma)$
$\quad ctx\_sub := \emptyset$
$\quad$ **while** $obuf_r \neq \emptyset$ **do**
(d) $\quad\quad$ pick σ' from $obuf_r$ and $obuf_r := obuf_r \setminus \{\sigma'\}$
$\quad\quad ctx\_sub := ctx\_sub \cup \mathsf{lconfig}(C_{root}, R \setminus \{r\}, \sigma')$
$\quad$ **return** ctx_sub

dynamic MCS. We start with a particular context in the system and gradually invoke some neighbors to get further solutions.[3]

4.1 Basic Algorithm

Given a dynamic MCS M and a starting context C_{root}, the algorithm lconfig presented in this section aims at enumerating all possible context substitutions that can lead to a binding for M, in a distributed way. It mutually calls an algorithm invoke_neighbors and makes use of the following primitives:

- a function get_contexts(σ), which takes a context substitution σ (containing substitutions of form $X \mapsto C$) as input and returns the set of contexts C used in σ.
- a DFS subroutine bind_rule, which given an s-bridge rule r as input consults the matchmaker MatchMaker and returns all context substitutions for the non-ordinary s-bridge atoms of r.

The algorithm lconfig has several parameters: the context C_{root} where the configuration started, the set R of s-bridge rules left to be bound, and the context substitution σ built up so far.

Intuitively, in a context C_k, lconfig first utilizes bind_rule in a DFS manner to enumerate all possible context substitutions for the s-bridge atoms in sbr_k (Step (b)). When this is done, in Step (a) it only refers to newly chosen contexts via a set $\mathcal{C}_{new}$ and calls invoke_neighbors to get the context substitutions of all members in $\mathcal{C}_{new}$.

The algorithm invoke_neighbors has the same parameters C_{root} and σ as lconfig, and carries in addition a set N of newly chosen neighbors of C_k where local configuration needs to be done. The algorithm first picks a neighbor C_j and calls lconfig at

[3] In centralized settings, one might instead compute substitutions by making use of more standard declarative solvers, e.g., such as ASP solvers with external information access.

Algorithm 2: invoke_neighbors(C_{root}, N, σ) at C_k

Input: C_{root}: root context, N: set of neighbors, σ: context substitution
Output: context substitutions for all neighbors of C_k

(e) **if** $N = \emptyset$ **then return** $\{\sigma\}$
else
(f) pick C_j from N, and $obuf_{C_j} := C_j$.lconfig$(C_{root}, sbr_j, \sigma)$
$ctx\_sub := \emptyset$
while $obuf_{C_j} \neq \emptyset$ **do**
(g) pick σ' from $obuf_{C_j}$ and $obuf_{C_j} := obuf_{C_j} \setminus \{\sigma'\}$
(h) $N' := N \setminus (\text{get\_contexts}(\sigma') \cup \{C_j\})$
$ctx\_sub := ctx\_sub \cup \text{invoke\_neighbors}(C_{root}, N', \sigma')$
return ctx_sub

this context (Step (f)) to get all context substitutions updated with local substitutions for C_j, stored in $obuf_{C_j}$. Then, in Step (g), it picks each substitution from $obuf_{C_j}$ and continues invoking the remaining contexts in N. Note that in Step (h), the set of remaining neighbors to invoke is recomputed in N', as some of the contexts in N might already be chosen by the call to C_j and thus they are already invoked.

When all invocations of neighbors have finished, the substitution computed at this point is returned and is treated by lconfig either as an intermediate result for the context that invoked it, or as the final result for the user.

Example 9 Take the setting from Example 8 and run bind_rule over the rule body with a starting empty substitution. The call is bind_rule$(B, \emptyset)$ in which $B = \{(X_1 : [pos_1]), (Y_1 : [neg_1])\}$. Assume that the first s-bridge atom chosen at Step (i) is $sb = (X_1 : [pos])$. A call MatchMaker$([pos_1], C_1)$ to the matchmaker returns $N = \{C_2, C_3\}$. The routine then tries all possibilities to bind sb and works recursively to bind the rest of the body. For example, if it chooses to bind X_1 to C_2, then the next call will be bind_rule$(\{(Y_1 : [neg_1])\}, \{X_1 \mapsto C_2\})$ which returns $\{\{X_1 \mapsto C_2, Y_1 \mapsto C_3\}\}$ as the set of all context substitutions in which X_1 is mapped to C_2. The binding continues with $X_1 \mapsto C_3$ and in the end, we get two context substitutions, namely $\{X_1 \mapsto C_2, Y_1 \mapsto C_3\}$ and $\{X_1 \mapsto C_3, Y_1 \mapsto C_2\}$.

Example 10 This example illustrates the run of lconfig and invoke_neighbors on a dynamic MCS from Example 4 with poolsize of contexts $n = 3$. Starting from C_1 we can pick one s-bridge rule from the non-empty set of s-bridge rules at (c), say the first one from Example 4. According to Example 9, the subroutine bind_rule returns a set of 2 possible context substitutions. Let us pick $\sigma = \{X_1 \mapsto C_2, Y_1 \mapsto C_3\}$ from this set and continue calling lconfig for the last rule. This gives 2 possible extensions of σ, one of which extends σ to $\{X_1 \mapsto C_2, Y_1 \mapsto C_3, Z_1 \mapsto C_2, T_1 \mapsto C_3\}$.

Having this context substitution carried to the next recursive call of lconfig, we reach the point where $R = \emptyset$, get $\mathcal{C}_{new} = \{C_2, C_3\}$, and continue calling lconfig at C_2 or C_3. The algorithm proceeds and in the end, we get a number of context substitutions, one is $\{X_1 \mapsto C_2, Y_1 \mapsto C_3, Z_1 \mapsto C_2, T_1 \mapsto C_3, X_2 \mapsto C_1, Y_2 \mapsto C_3, Z_2 \mapsto C_3, T_2 \mapsto C_1, X_3 \mapsto C_2, Y_3 \mapsto C_1, Z_3 \mapsto C_1, T_3 \mapsto C_2\}$. This substitution yields the MCS system in Example 3.

Algorithm 3: bind_rule(I, B, σ) at C_k

Input: I: set of inequality atoms, B: set of s-bridge atoms, σ: context substitution
Output: substitutions for B

(i) **if** $\exists a = (X : P)$ *non-ordinary in* B **then**
  $N :=$ MatchMaker(P, C_k) // N: set of potential neighbors
  if $\nexists (X \mapsto C)$ *in* σ **then**
    $dup := \{C_i \in N \mid (Y \mapsto C_i) \in \sigma \wedge (X \neq Y) \in I\}$
    $N := N \setminus dup$
    $ctx\_sub := \emptyset$
    while $N \neq \emptyset$ **do**
(j) choose a context C_j from N, and $N := N \setminus \{C_j\}$
      $ctx\_sub := ctx\_sub \cup$ bind_rule$(I, B \setminus \{a\}, \sigma \cup \{X \mapsto C_j\})$
    return ctx_sub
  else if $\sigma(X) \in N$ **then return** bind_rule$(I, B \setminus \{a\}, \sigma)$
  else return $\emptyset$
else return $\{\sigma\}$

When the pool size gets large, enumerating all bindings for each bridge rule and all bridge substitutions becomes infeasible. A practical approach would be to compute only a small number of bindings for each rule, and also just a few substitutions at each context. For the remainder of this section, let us use b and n to denote corresponding limits.

We have presented the basic algorithm for enumerating all possible context substitutions of a dynamic MCS M w.r.t. a context C_k in M with which M can be bound to original MCS. To keep it simple, in steps (c), (d), (f), (g), (i), and (j), we nondeterministically pick either a rule, a context substitution, or a context as no supporting information is provided. This leaves a lot of room for optimization. Furthermore, we did not mention how to deal with irregular cases such as when the matchmaker returns no potential neighbor, or the size of the partial MCS has passed some boundary; furthermore, no caching has been foreseen.

In the following subsection, we discuss different heuristics to enhance the search process when more support information is available, so that the context substitutions will be returned in some quality driven order. After that, we briefly describe a strategy for cutting off when reaching a size boundary, hence a possibility to tolerate partial bindings.

4.2 Quality-Driven Local Configuration

Quality for the topology (Q_T). Our experimental results reveal that evaluating equilibria of MCS in general does not scale up to very large systems, and Dao-Tran et al. [2010] and Bairakdar, Dao-Tran, Eiter, Fink, and Krennwallner [2010] showed that limitations on some specific topologies such as the diamond topology exist. Hence, one of the purposes for configuration is to restrain the size/topology of the resulting system to some boundary, e.g., by trying to reuse as many contexts from the local

configuration of the parent as possible; or by trying to avoid troublesome topological properties, such as ones having *join contexts*, i.e., contexts C_i which are accessed from different contexts C_j and $C_{j'}$ which in turn are accessed (possibly by intermediate contexts) from a single context C_k, or cycles.

For this purpose, the selection of neighbors (Step (j)) is crucial. To support a context C_k with more information for this task, we do a *one-step look-ahead* at all of its potential neighbors. In general, looking ahead into a potential neighbor C_i can give back any information that C_i is able to infer from its own knowledge and information provided by the matchmaker. In this paper, our setting allows the look-ahead to return the number of s-bridge atoms in C_i, denoted by nba_i.

We define in the following different heuristic possibilities of the topological quality function to reflect attempts to have the resulting MCS in some restricted shape (the smaller the value of the function, the better is the quality).

Assume that having started the configuration from context C_{root}, we are now doing local configuration at context C_k, choosing a binding for a schematic bridge belief $[p]$, considering the possible match (p, q) to a context C_i. As the context substitution σ is carried along, one can easily extract the set of chosen contexts so far, which is denoted here by $\mathcal{C}$. Consider the following topological quality functions:

(H1) $quality_{i,k} = nba_i$: with this function, we prefer potential neighbors with fewer s-bridge atoms.

(H2) $quality_{i,k} = \begin{cases} 0 & \text{if } C_i \in \mathcal{C} \\ 1 & \text{otherwise.} \end{cases}$

This function gives priority to contexts which are already chosen, hence to keep the size of the resulting system small. On the other hand, this tends to introduce cycles.

One can imagine more complicated quality functions; for instance, to take the topology of the system built up so far into account in order to avoid cycles or join contexts. To this end, one must transfer not only the substitution σ between contexts, but also the system topology (i.e., the respective graph).

Along the same lines, for Step (c) (resp., (i)) one can define quality functions, for instance based on some syntactic criteria combined with some history information, to provide an heuristic ranking for choosing the next rule (resp., the next non-ordinary s-bridge atom).

Quality for bindings of a schematic rule (Q_S). This type of quality measures the closeness between the bindings and the intended meaning of the schematic rule based on the matching quality of each single s-bridge atom. Notice that after getting the schematic substitution η from the matchmaker, θ is determined by the context substitution σ. Each realization of σ gives us a possibility to bind the schematic rules. What we need is a means to compare these possibilities. To make it generic, we define the quality function ρ of a bound rule $r' \in r\sigma$ of a schematic rule r of form (3) as follows:

$$\rho(r') = \mathrm{op} \left\{ \begin{array}{l} \alpha \mid sb = (X : P) \in B(r) \wedge \\ \quad (C_j : b) \in \theta(sb) \wedge (bel(P), b) \in \eta(C_i, C_j) \wedge \\ \quad (C_j : b) \in B(r') \wedge f_M(bel(P), b) = \alpha \end{array} \right\}.$$

Basically, we take the measure of similarity of all bindings used to construct r' and apply an operator op on top. Here, op is generic and can be instantiated to any operator for a specific use. For example, two plausible options are (i) $\mathrm{op} = \min$, and (ii) $\mathrm{op} = \mathrm{avg}$.

Roughly, in case (i), following an overly cautious approach, the quality of the whole binding is determined as the minimal quality of all matches of the s-bridge atoms in its body. In a different approach, case (ii) takes all matches into account and respects a contribution of each match in the overall quality of the rule. Depending on different philosophies to establish the overall quality of a binding that is based on bindings of each single schematic bridge atom, one can provide more complicated operators and plug them into this scheme.

To benefit from this quality, one can sort the output buffer $obuf_r$ (resp., $obuf_{C_j}$) according to Q_S in Step (d) (resp., (g)), and then pick the best context substitution up to this point to continue with.

Combined quality (Q_C). The two types of quality functions above look into two aspects of the resulting system, namely the (Q_T) topology and the (Q_S) similarity in meaning of the bindings. To exploit the latter, one needs to enumerate all possible bindings for a rule.

When we have a limit on the number of solutions (see discussion above), it is very important to approximate Q_T when choosing a binding, since then one cannot compute all bindings of a rule, and then sort them.

To this end, we modify Q_T in a way that it also takes care of the quality of the match. Intuitively, when two potential contexts are equally ranked by Q_T, one looks at the quality of the match, i.e., approximating Q_S, to rank them. More specifically, the heuristic functions H1 and H2 are changed to:

(H3) $quality_{i,k} = nba_i - \alpha$, and

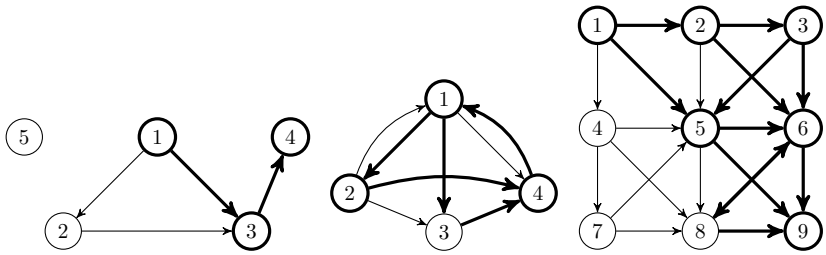
(H4) $quality_{i,k} = \begin{cases} -\alpha & \text{if } C_i \in \mathcal{C} \\ 2 - \alpha & \text{otherwise,} \end{cases}$

where α is the quality of the match being considered to bind the current s-bridge atom.

4.3 Dealing with Irregular Cases

In practice, it is convenient for the user to have the possibility to specify an upper bound for the size of the resulting system. However, a full substitution respecting the given limit may not always exist. A flexible approach to deal with such a situation, rather than to increase the bound, is to *cut off* when reaching the boundary and to tolerate partial answers.

By cutting off, more precisely we mean to remove all unbound negative s-bridge atoms from s-bridge rules, and remove all s-bridge rules with unbound positive bridge atoms. Intuitively, this amounts to consider a system where any further contexts that would exist for binding are considered to return empty belief sets (and thus the respective bridge atoms are pre-evaluated accordingly). Note that one also needs to undo the on-going substitutions for such s-bridge rules, and this might trigger the cancellation of substitutions in a backward manner: since once a context is not used anymore for



(a) Random (b) Rake (c) 3×3 Grid

Figure 1: Benchmark topologies and possible configurations (emphasized)

instantiating other contexts, it is not needed and the part of the substitution w.r.t. this context should be removed from the final result.

Another case in which cutting off might be used is when substitutions do not exist due to non-existent matchings, i.e., when the matchmaker does not return any match for a schematic constant. We can apply the same strategy as above, i.e., remove the corresponding s-bridge atom if it appears in the negative body of an s-bridge rule, or remove the whole s-bridge rule if the s-bridge atom is in its positive body.

The cutoff is in fact easy to implement. One can create a dummy context C_0 which has only a single belief $dumb$ with the unique acceptable belief set $\emptyset$ (i.e., intuitively $dumb$ is $false$, and all beliefs in every other context are matchable to $dumb$. Then cutting off as discussed above is simply achieved by matching unbound s-bridge atoms to $(C_0 : dumb)$.

5 Implementation and Experimental Results

In this section, we report on some initial experiments with a prototype implementation of our MCS configuration algorithm. Full details of the implementation and the experiments are available on the web.[4]

Our implementation is written in C++ and uses a simple realization of MatchMaker, in which matchmaking results are statically given rather than dynamically computed. The respective matching information is stored in a text file, which is loaded into each context at start up time. During configuration, a call to the matchmaker is then simply performed by posing a query to the respective storage in the context.

For our experiments, we used a host system having an Intel Core 2 Duo 2.4GHz processor with 4GB RAM, running Mac OS X. We have tested three types of topologies, called random, rake, and grid, respectively, which model different interlinkage patterns (see Fig. 1 for exemplary instances). The *random* topology is generated by deciding whether there exist potential matches from context C_i to context C_j with a probability of 0.5; this leads to a rather dense system as random topologies have approximately half of the edges of a complete graph. Fig. 1a shows an example with five contexts, for which C_1, C_3, and C_4 has been chosen in the configuration. The *rake* topology requires that every context C_i has potential matches to all contexts C_j where $j > i$; furthermore, for each context C_i with $i > 1$, we randomly pick $\lceil i/3 \rceil$ distinct

[4] `http://www.kr.tuwien.ac.at/research/systems/dmcs/`.

context(s) C_k where $k < i$, and with a probability of 0.7, place potential matches from C_i to C_k. The example instance in Fig. 1b has a configuration using all contexts C_1 to C_4, which forms a system with two cycles. And finally, in instances of the *grid* topology having size $m \times n$, each context has potential matches to its adjacent contexts on the next row/column, a diagonally adjacent context on the next row and next column; and for contexts on the last row/column, diagonally adjacent backward/forward contexts (see Fig. 1c).

For each topology, we experimented with dynamic MCS having a pool of 100 or 200 contexts. Each context contains a local knowledge base as in Example 2 and a set of bridge rules which is a subset of $sbr_i \cup \{r_i\}$ where sbr_i is from Example 4 and $r_i = inc_i \vee dec_i \leftarrow (X_i : [pos]), (X_i : [neg])$, in order to have a varying number of s-bridge atoms in contexts. Furthermore, the limits on the number of substitutions and bindings were set to $n = 3$ and $b = 8$, respectively.

The results of running our basic algorithm (no heuristics) and the four heuristics H1–H4 are shown in Tables 2–4. There, the test names are of the form $Name(ps, d)$ with poolsize ps and density d is the number of possible connections between contexts of the dynamic MCS instance. The measurements of the output include the running time of the algorithm, the size and the density (i.e., the number of connections between contexts) of the resulting MCS, and the average similarity of all matches used for instantiation.

As a general observation, we can see that most of the time the heuristics improved the result in different aspects, either the size respectively density, or the average quality, or both. The heuristics H1 and H2 concentrate on minimizing the size of the system, which was strongly confirmed in case of the rake topology. Here, the basic algorithm yielded large configurations: the default ordering for the next neighbor does not take advantage of back edges, because it always picks first a context with a higher index. Hence, for some first results, the configuration has to wait until the last context is bound; only then it can close the system.

The same behavior of the basic algorithm could be observed in the case of the grid topology: it always took the whole set of contexts for instantiation. On the other hand, when a heuristics was applied, the system size was often reduced by almost 50%, except for H2; this is because in this restricted structure, there are no backward edges at intermediate contexts, hence the effort to close the system can only take effect when contexts at the last row or column are involved in the instantiation.

However, since the random topology has no structure, the heuristics H1 and H2 behave similarly to running the algorithm without heuristics. The former even returned a slightly bigger system and many times one of worse quality.

On the other hand, the heuristics H3 and H4 combine the topology and quality criteria in order to intuitively gain an improvement; this indeed shows up in the experimental results. For the random topology, H3 dominated the result in quality; this was achieved by a considerably larger configuration, which is explained by the fact that heuristics H3 branches out to promising contexts for quality improvements, rather than looking at already chosen neighbors. In the other cases, either H3 or H4 gave the best quality in all the tests.

Moreover, the trade-off between the number of contexts and the quality of the configuration can be seen in comparing heuristics H3 and H4. While the former approximates the system size based on the number of bridge atoms, the latter has a more

Table 2: Experimental results for random topology

Test name	Heuristics	Running time (secs)	Size	Density	Quality
Random (100, 4974)	None	0.22	5	10	0.604
	H1	6.16	9	18	0.450
	H2	0.18	4	8	0.605
	H3	2.39	26	54	0.975
	H4	1.42	8	16	0.859
Random (100, 4886)	None	0.18	7	14	0.643
	H1	1.01	9	18	0.603
	H2	0.19	6	12	0.584
	H3	6.65	17	34	0.975
	H4	0.36	8	16	0.703
Random (200, 19953)	None	0.62	6	12	0.584
	H1	2.23	7	14	0.480
	H2	0.40	6	12	0.588
	H3	8.92	44	91	0.996
	H4	0.82	7	14	0.852
Random (200, 19896)	None	1.02	6	12	0.510
	H1	0.84	6	12	0.605
	H2	1.12	6	12	0.510
	H3	18.88	37	76	0.999
	H4	1.29	7	14	0.794

Table 3: Experimental results for rake topology

Test name	Heuristics	Running time (secs)	Size	Density	Quality
Rake (100, 5575)	None	10.61	99	198	0.529
	H1	1.00	35	70	0.530
	H2	1.01	10	20	0.500
	H3	7.65	28	57	0.983
	H4	0.48	10	20	0.908
Rake (100, 5597)	None	22.76	100	200	0.557
	H1	1.01	26	52	0.514
	H2	1.29	10	20	0.498
	H3	0.96	22	45	0.937
	H4	1.51	9	18	0.867
Rake (200, 22579)	None	170.10	197	394	0.541
	H1	17.11	73	146	0.507
	H2	0.82	9	18	0.518
	H3	64.24	59	119	0.999
	H4	0.88	5	11	0.790
Rake (200, 22510)	None	277.04	199	398	0.550
	H1	4.84	54	108	0.579
	H2	1.65	9	18	0.520
	H3	4.40	36	72	0.994
	H4	0.74	9	18	0.746

Table 4: Experimental results for grid topology

Test name	Heuristics	Running time (secs)	Size	Density	Quality
Grid (10×10, 279)	None	1.82	100	198	0.540
	H1	1.60	66	130	0.568
	H2	4.02	100	198	0.545
	H3	2.87	74	154	0.630
	H4	2.02	64	128	0.736
Grid (20×5, 274)	None	2.52	100	198	0.562
	H1	1.84	57	112	0.565
	H2	5.05	82	162	0.557
	H3	2.38	67	134	0.623
	H4	2.04	52	104	0.724
Grid (25×4, 270)	None	3.44	100	198	0.519
	H1	1.81	50	98	0.534
	H2	8.64	100	198	0.527
	H3	1.46	50	98	0.591
	H4	1.94	55	109	0.623
Grid (20×10, 569)	None	5.32	200	398	0.544
	H1	6.90	130	258	0.541
	H2	25.62	200	398	0.551
	H3	9.10	152	323	0.631
	H4	9.86	104	206	0.677
Grid (25×8, 566)	None	6.78	200	398	0.563
	H1	10.60	142	282	0.563
	H2	31.54	200	398	0.567
	H3	2.61	87	178	0.623
	H4	10.15	83	164	0.705
Grid (40×5, 554)	None	8.44	200	398	0.557
	H1	5.10	102	202	0.577
	H2	29.46	162	322	0.570
	H3	5.35	103	215	0.629
	H4	5.88	91	184	0.686

direct approach by looking at the chosen contexts. Hence H4 usually ended up with a smaller system, but not always with one of better quality. On the other hand, H3 always returned a bigger system compared to standard results, but interestingly it could not beat H4 in the case of grid topology, because this rigid structure gives H3 no chance to trade system size for quality.

On average, H4 appeared to have the best balance of all result aspects together: it has fast running time, almost the best quality, and acceptable system size/density.

To see the effects of inequality atoms in s-bridge rules, we have also run our algorithm on the same test cases with these special atoms removed. In general, the trends and the relationships between the basic algorithm and its heuristic extensions do not change (see Appendix A). The only difference now is that the size/density of the resulting systems gets smaller, as a single context can be used to bind two different context holders in one rule, and the algorithm tries to exploit this property to come to the answers as soon as possible.

We have carried out further experiments with a social-groups topology that are not reported here. This topology forms groups of contexts with full potential connections,

and there exist some rare potential connections from one group to another; in some sense this models loosely interconnected communities. The outcome of our configuration resembled typical human behavior of group formation in such a setting quite well, as the resulting MCS tended to contain contexts from a single group only.

6 Related Work and Conclusion

We have presented the framework of dynamic multi-context systems, which extend ordinary multi-context systems (MCS) with so-called schematic bridge rules in order to allow for open environments, in which concrete contexts are taken at run time to form an ordinary MCS by instantiating the schematic bridge rules. We have developed a distributed algorithm for instantiating such dynamic MCS to ordinary MCS, implemented this algorithm in a prototype system, and have shown some benchmark results that compare different heuristics for configuring dynamic MCS.

6.1 Related Work

The problem that we considered in this paper shares some similarities with configuration in Multi-Agent Systems using matchmaking. In this setting [Sycara et al., 2002], provider agents advertise their capabilities to middle agents; requester agents do not directly go to a provider but first ask some middle agent whether it knows of providers with desired capabilities; the middle agent matches the request against the stored advertisements and returns the information about appropriate providers to the requester. In our setting, the matchmaker plays the role of the middle agent. A context has both roles, it is seen as a requester when being instantiated, and as a provider when being used to instantiate s-bridge atoms from other contexts.

A configuration problem for multi-agent system that is in a sense orthogonal respectively complementary to matchmaking is coalition formation. Here, the problem is the assembly of a group of agents for cooperation in order to get some task down (assuming that the agents already know that cooperation is possible) [Sandholm, 1999]. However, this problem is only remotely related to our configuration problem. Agents have goals and intentions, and decide their participation in a coalition based on utility and reward in a rational manner. This leads in interaction with other agents to complex behaviors, which may be studied using game-theoretic methods and tools. Contexts instead lack such goal and reward orientation, and offer in an altruistic manner information exchange in order to enable the assembly of an MCS. Thus, from a coalition formation point of view, the MCS configuration problem is trivial. The problems gets more complicated if constraints are imposed (e.g., on the solution size or quality), but there is still a difference: at no point, some context may decide not to participate in an MCS as it concludes its payoff is insufficient, or it is being cheated.

Naturally related to dynamic MCS are peer-to-peer (P2P) systems. However, in typical models such as the Peer-Grid [Aberer et al., 2002], a global system semantics does not play a role: peers are strictly localized and can join/leave the system at anytime. Our approach, on the other hand, aims at global model building for an ordinary MCS that is dynamically constructed, where the first step is instantiation, and then the (distributed) evaluation kicks in [Dao-Tran et al., 2010]; this tacitly assumes that no

relevant contexts disappear during configuration and evaluation of the configured system. We note that also [Calvanese, Giacomo, Lenzerini, and Rosati, 2004] proposed a global model semantics for P2P systems, which is based on epistemic logic, and presented a distributed algorithm for query answering. This algorithm evaluates P2P mappings dynamically, but no system configuration like in our approach is performed. Similarly, [Bikakis and Antoniou, 2010] considered distributed query answering in a given P2P system of contexts, but under preferences using an argumentation based approach; however, no dynamic configuration in a potentially open environment is performed.

6.2 Issues for future research

While we have introduced in this paper dynamic MCS to accommodate open environments and we provided an algorithm for run time configuration, several issues remain for future work.

On the foundational side, a study of the computational complexity of dynamic MCS, and in particular of the configuration problem, could reveal important insight into computational resources needed to solve this problem, and may help to identify classes of systems for which it is efficiently solvable; here, the distribution and possible parallelism are interesting aspects. Furthermore, an improvement of the configuration algorithm, and in particular a deep investigation into heuristics would be an interesting task. Another aspect related to this is linkage cost. The size of a configuration is a crude measure of such cost, which clearly can be refined, taking, e.g., besides the topology also the cost (or value) of accessing particular beliefs into account.

On the implementation side, an obvious task is the implementation of a full-fledged configuration system that includes rich matchmaking, e.g., as in LARKS [Sycara et al., 2002] instead of just hard-coded matches. Finally, another issue are applications of dynamic MCS. The student example in the Introduction suggests to consider possible applications in social group formation, complementing e.g., recent work in social Answer Set Programming [Buccafurri, Caminiti, and Laurendi, 2008, Buccafurri and Caminiti, 2008]. Another, less mundane area is configuration of small heterogeneous information systems, in which generic components (e.g., some domain ontologies, some decision component, and some fact base) must be suitably instantiated, given various possibilities. Here matchmaking may play an important role, e.g., if aspects like different levels of abstraction in the context knowledge bases should be handled. The usage of logic-based matchmaking approaches (cf. the work of Noia, Sciascio, and Donini [2007]), in combination with other techniques, might here be worthwhile to consider. In particular, configuration of small systems in mobile environments, where openness is a natural requirement, to further the use of multi-context systems in ambient intelligence [Antoniou et al., 2010, Bikakis and Antoniou, 2010] would be interesting.

References

Karl Aberer, Magdalena Punceva, Manfred Hauswirth, and Roman Schmidt. Improving Data Access in P2P Systems. *IEEE Internet Computing*, 6(1):58–67, 2002. doi: 10.1109/4236.978370.

Grigoris Antoniou, Constantinos Papatheodorou, and Antonis Bikakis. Reasoning about Context in Ambient Intelligence Environments: A Report from the Field. In Lin and Sattler [2010], pages 557–559. URL http://aaai.org/ocs/index.php/KR/KR2010/paper/view/1209.

Seif El-Din Bairakdar, Minh Dao-Tran, Thomas Eiter, Michael Fink, and Thomas Krennwallner. Decomposition of Distributed Nonmonotonic Multi-Context Systems. In Tomi Janhunen and Ilkka Niemelä, editors, *12th European Conference on Logics in Artificial Intelligence (JELIA 2010), Helsinki, Finland, September 13-15, 2010*, volume 6341 of *LNAI*, pages 24–37. Springer, September 2010. doi: 10.1007/978-3-642-15675-5_5. URL http://www.kr.tuwien.ac.at/staff/tkren/pub/2010/jelia2010-decompmcs.pdf.

Antonis Bikakis and Grigoris Antoniou. Defeasible Contextual Reasoning with Arguments in Ambient Intelligence. *IEEE Transactions on Knowledge and Data Engineering*, 22(11):1492–1506, 2010. doi: 10.1109/TKDE.2010.37.

Gerhard Brewka and Thomas Eiter. Equilibria in Heterogeneous Nonmonotonic Multi-Context Systems. In Robert C. Holte and Adele Howe, editors, *22nd AAAI Conference on Artificial Intelligence (AAAI'07)*, pages 385–390. AAAI Press, 2007. URL http://www.aaai.org/Papers/AAAI/2007/AAAI07-060.pdf.

Gerhard Brewka, Floris Roelofsen, and Luciano Serafini. Contextual default reasoning. In Manuela M. Veloso, editor, *International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 268–273. AAAI Press, 2007. URL http://ijcai.org/Past%20Proceedings/IJCAI-2007/PDF/IJCAI07-041.pdf.

Francesco Buccafurri and Gianluca Caminiti. Logic programming with social features. *Theory and Practice of Logic Programming*, 8(5-6):643–690, 2008. doi: 10.1017/S1471068408003463.

Francesco Buccafurri, Gianluca Caminiti, and Rosario Laurendi. A logic language with stable model semantics for social reasoning. In M. Garcia de la Banda and E. Pontelli, editors, *24th International Conference on Logic Programming (ICLP'08), Udine, Italy, December 9-13 2008*, volume 5366 of *LNCS*, pages 718–723. Springer, 2008. doi: 10.1007/978-3-540-89982-2_64.

Diego Calvanese, Guiseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Logical Foundations of Peer-To-Peer Data Integration. In *23rd ACM Symposium on Principles of Database Systems (PODS'04)*, pages 241–251. ACM, 2004. doi: 10.1145/1055558.1055593.

Minh Dao-Tran, Thomas Eiter, Michael Fink, and Thomas Krennwallner. Distributed Nonmonotonic Multi-Context Systems. In Lin and Sattler [2010], pages 60–70. URL http://aaai.org/ocs/index.php/KR/KR2010/paper/view/1249.

Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3–4):365–385, 1991. doi: 10.1007/BF03037169.

Fausto Giunchiglia and Luciano Serafini. Multilanguage Hierarchical Logics or: How we can do Without Modal Logics. *Artificial Intelligence*, 65(1):29–70, 1994. doi: 10.1016/0004-3702(94)90037-X.

Fangzhen Lin and Ulrike Sattler, editors. *12th International Conference on the Principles of Knowledge Representation and Reasoning (KR 2010), Toronto, Canada, May 9-13, 2010*, May 2010. AAAI Press. URL `http://www.aaai.org/Library/KR/kr10contents.php`.

Lucantonio Ghionna Michael Fink and Antonius Weinzierl. Relational Information Exchange and Aggregation in Multi-Context Systems. In James Delgrande and Wolfgang Faber, editors, *11th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2011), Vancouver, BC, Canada, 16-19 May, 2011)*, LNCS. Springer, 2011. to appear.

George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11): 39–41, 1995.

Tommaso Di Noia, Eugenio Di Sciascio, and Francesco M. Donini. Semantic Matchmaking as Non-Monotonic Reasoning: A Description Logic Approach. *J. Artif. Intell. Res.*, 29:269–307, 2007. doi: 10.1613/jair.2153.

Elth Ogston and Stamatis Vassiliadis. Local Distributed Agent Matchmaking. In Carlo Batini, Fausto Giunchiglia, Paolo Giorgini, and Massimo Mecella, editors, *9th International Conference on Cooperative Information Systems (CoopIS'01)*, volume 2172 of *LNCS*, pages 67–79. Springer, 2001. doi: 10.1007/3-540-44751-2_7.

F. Roelofsen and L. Serafini. Minimal and Absent Information in Contexts. In L. Pack Kaelbling and A. Saffiotti, editors, *19th International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 558–563. Morgan Kaufmann, 2005. URL `http://www.ijcai.org/papers/1045.pdf`.

Tuomas Sandholm. Distributed rational decision making. In Gerhard Weiss, editor, *Multiagent Systems – A Modern Approach to Distributed Artificial Intelligence*, chapter 5, pages 201 – 258. MIT Press, 1999.

Katia P. Sycara, Seth Widoff, Matthias Klusch, and Jianguo Lu. Larks: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace. *Autonomous Agents and Multi-Agent Systems*, 5(2):173–203, 2002. doi: 10.1023/A:1014897210525.

A Appendix: Experimental results (no inequalities)

Table 5: Experimental results for grid topology, without inequalities

Test name	Heuristics	Running time (secs)	Size	Density	Quality
Grid (10×10, 279)	None	0.13	19	18	0.558
	H1	0.13	16	16	0.553
	H2	0.12	19	18	0.558
	H3	0.73	20	25	0.589
	H4	0.10	17	17	0.592
Grid (20×5, 274)	None	0.17	24	23	0.597
	H1	0.19	28	28	0.547
	H2	0.16	24	23	0.597
	H3	1.10	30	37	0.598
	H4	0.16	24	23	0.658
Grid (25×4, 270)	None	0.22	28	27	0.529
	H1	0.22	31	31	0.579
	H2	0.20	28	27	0.529
	H3	1.42	37	50	0.623
	H4	0.25	34	34	0.616
Grid (20×10, 569)	None	0.20	29	28	0.480
	H1	0.21	30	30	0.459
	H2	0.20	29	28	0.480
	H3	1.54	49	61	0.605
	H4	0.20	26	25	0.712
Grid (25×8, 566)	None	0.25	32	31	0.566
	H1	0.26	32	32	0.592
	H2	0.25	32	31	0.566
	H3	1.73	35	41	0.638
	H4	0.91	28	28	0.617
Grid (40×5, 554)	None	0.49	44	43	0.538
	H1	0.85	46	46	0.535
	H2	0.37	44	43	0.538
	H3	2.02	53	65	0.628
	H4	0.53	53	52	0.630

Table 6: Experimental results for random topology, without inequalities

Test name	Heuristics	Running time (secs)	Size	Density	Quality
Random (100, 4974)	None	0.39	2	2	0.567
	H1	0.46	3	3	0.638
	H2	0.37	2	2	0.567
	H3	2.61	23	44	0.981
	H4	0.37	4	4	0.764
Random (100, 4886)	None	0.43	3	3	0.573
	H1	0.47	3	3	0.573
	H2	0.15	3	3	0.573
	H3	9.74	17	33	0.977
	H4	0.46	4	4	0.833
Random (200, 19953)	None	0.38	2	2	0.711
	H1	0.78	5	5	0.415
	H2	0.10	2	2	0.711
	H3	8.13	42	80	0.996
	H4	0.13	3	3	0.631
Random (200, 19896)	None	0.39	2	2	0.483
	H1	0.46	2	2	0.467
	H2	0.43	2	2	0.483
	H3	36.58	36	70	0.999
	H4	0.20	3	3	0.686

Table 7: Experimental results for rake topology, without inequalities

Test name	Heuristics	Running time (secs)	Size	Density	Quality
Rake (100, 5575)	None	3.08	91	91	0.521
	H1	0.52	32	32	0.533
	H2	0.34	6	6	0.367
	H3	8.46	27	49	0.983
	H4	0.08	5	5	0.744
Rake (100, 5597)	None	3.15	90	90	0.568
	H1	0.25	22	22	0.508
	H2	0.69	5	5	0.445
	H3	0.82	20	38	0.951
	H4	0.16	3	3	0.682
Rake (200, 22579)	None	11.70	174	174	0.551
	H1	1.09	66	66	0.491
	H2	0.42	4	4	0.494
	H3	39.95	57	111	0.999
	H4	0.64	4	4	0.815
Rake (200, 22510)	None	6.04	173	173	0.541
	H1	0.81	48	48	0.593
	H2	2.91	4	4	0.481
	H3	4.47	36	72	0.993
	H4	0.55	3	3	0.667