An Agent-Based Framework for Context-Driven Interpretation of Symbols in Diagrammatic Sketches

Giovanni Casella[1,2], Gennaro Costagliola[2], Vincenzo Deufemia[2], Maurizio Martelli[1], Viviana Mascardi[1]

[1] Dipartimento di Informatica e Scienze dell'Informazione – Università di Genova
Via Dodecaneso 35, 16146, Genova, Italy, {casella,martelli,mascardi}@disi.unige.it

[2] Dipartimento di Matematica e Informatica – Università di Salerno
Via Ponte don Melillo, 84084 Fisciano(SA), Italy, {gcostagliola,deufemia}@unisa.it

Abstract

Parsing hand-drawn diagrams is a definitely complex recognition problem. The input drawings are often intrinsically ambiguous, and require context to be interpreted in a correct way. Many existing sketch recognition systems avoid this problem by recognizing single segments or simple geometric shapes in a stroke. However, for a recognition system to be effective and precise, context must be exploited, and both the simplifications on the sketch features, and the constraints under which recognition may take place, must be reduced to the minimum.

In this paper we present an agent-based framework for context-driven interpretation of symbols in diagrammatic sketches that heavily exploits contextual information for ambiguity resolution. Agents manage the activity of low-level hand-drawn symbol recognizers, that may be heterogeneous for better adapting to the characteristics of each symbol to be recognized, and coordinate themselves in order to exchange contextual information, thus leading to an efficient and precise interpretation of sketches.

1. Introduction

Recognition of hand-drawn diagrams is a very active research field, since it finds a natural application in a wide range of domains, such as engineering, software design, and architecture [6, 13, 16, 25]. However, it is a particularly difficult task since the symbols of a sketched diagram can be drawn by using a different stroke-order, -number, and -direction. The difficulties in the recognition process are often made harder by the lack of precision and by the presence of ambiguities in messy hand-drawn sketches. In fact, hand-sketched symbols are imprecise in nature: corners are not always sharp, lines are not perfectly straight, and curves are not necessarily smooth.

Usually, hand-drawn diagrams consist of parts whose meaning depends heavily on context. For example, a single line fragment could constitute the side of a box, or a connector between boxes, and its role could be disambiguated only by looking at neighboring fragments. This means that, when a recognized symbol is unique to a context, then the recognizer may exploit this symbol to determine the context and thereby resolve pending recognition ambiguities. The context can also be used to recover from low-level interpretation errors by reclassifying low-level shapes, obtaining significantly reduced recognition errors [3].

Besides this, sketch interpretation is always carried out by applying, to any symbol in the sketch, the same recognition approach. To the best of our knowledge, no framework exists that allows the adoption of different techniques for recognizing different symbols. Nonetheless, this might be a very useful feature for an effective recognition process, since each symbol shows its own peculiar characteristics, which make a particular recognition technique more or less suitable for it.

In this paper, we present an agent-based framework for context-driven interpretation of symbols in diagrammatic sketches. The reasoning process performed by the intelligent agents devoted to symbol recognition (*Symbol Recognition Agents*, SRA for short) and to the correct interpretation of the sketch (*Coordinator Agent*, CA for short), is based on the knowledge about the domain context, which is used for disambiguating the recognized symbols. At the lowest level of our framework, the symbols of the domain language are identified by applying suitable *Hand-Drawn Symbol Recognizers* (HDSRs, for short) to the interpretations of elementary strokes. The execution of these HDSRs is coordinated by SRAs. In spite of the differences among the existing HDSRs, several of them could be profitably integrated into our system. As long as there is one SRA that correctly integrates a set of HDSRs by managing their execution as well as data conversion issues, the actual implementation of the HDSRs and the approach to recognition that they adopt *do not matter*. For this reason, our framework has the potential to *seamlessly integrate symbols* that have been recognized by *heterogeneous* HDSRs.

An SRA exchanges contextual information by cooperating with other SRAs in the system. The contextual information obtained in this way is sent to the CA that solves possible conflicts and gives an interpretation of the sketch drawn so far. The CA is also able to reduce the number of active HDSRs for improving the performances of the system.

The paper is organized as follows. Section 2 motivates the use of multi-agent systems for sketch recognition. Section 3 describes the proposed framework, and Section 4 exemplifies our approach by describing the recognition system that we would obtain by integrating the HDSRs proposed in [10] and [17]. The related work is discussed in Section 5. Finally, conclusions and further research are discussed in Section 6.

2. Agents for sketch understanding

The AgentLink III Technology Roadmap [20] defines an agent *as:*

"a computer system that is capable of flexible autonomous action in dynamic, unpredictable, typically multi-agent domains."

According to [27], agents should be

1. *responsive*: they should perceive their environment and respond in a timely fashion to changes that occur in it;
2. *pro-active*: they should not simply act in response to their environment, but should exhibit opportunistic, goal-directed behaviour and take the initiative where appropriate;
3. *social*: they should be able to interact, when appropriate, with other artificial agents and humans in order to complete their own problem solving and to help others with their activities.

Another characterizing feature of agents is *situatedness:* the agent receives sensory input from its environment and that it can perform actions which change the environment in some way [19].

As far as sociality is concerned, it is now widely recognized that interaction is probably the most important single characteristic of nowadays' complex software. Two good reasons for agents to interact and eventually cooperate, are to solve conflicts [2, 7], and to disambiguate the interpretation of objects in some domain [11].

If we take the above features in mind while considering the problem of recognizing hand-drawn sketches exploiting contextual information, we soon realize that an architecture based on agents might be a proper solution.

In fact, the "virtual blank sheet" where the user draws represents a dynamic and unpredictable environment, and an "entity" devoted to recognizing a specific symbol of some language must be situated in it, in order to properly perceive the user's actions. Also, this entity must react to changes that take place in the virtual blank sheet, i.e., new strokes drawn by the user, have a complex long term goal, i.e, giving a correct interpretation to what the user is drawing, and operate in an autonomous way to reach this goal, since no explicit input or suggestions must be required to the user. Finally, although each single entity may be able to recognize one specific symbol of the language with a certain degree of confidence, by working alone, it cannot easily resolve ambiguities ("Is this symbol an arrow or a line?"), and conflicts ("In order to recognize my symbol, I am using a stroke that is also used by another entity; to which symbol does the stroke really belong?"). Thus, a social behavior is required to reach the final goal of each entity, that consists in overcoming conflicts and ambiguities, and providing the right interpretation of the sketch to the user. In the end, this "entity" must be responsive, pro-active, situated, autonomous, and social. In other words, it must be an intelligent agent.

3. Our agent based framework

The framework that we propose for context-driven interpretation of symbols is depicted in Fig. 1. It is composed of four modules:

Interface Manager. The Interface Manager allows the usage of the framework with generic sketch editors (not included in our framework). It is responsible for converting the information produced by the editor into a format compliant with the framework, and vice versa.

Stroke Interpreter. The Stroke Interpreter classifies the user strokes into a sequence of domain independent primitive shapes with attributes, which are stored in a shared repository. It receives the sequence of strokes drawn by a user from the Interface Manager. A stroke is represented by a sequence of points whose sampling density is dependent on the sketching speed. After a re-sampling stage, the strokes are segmented by

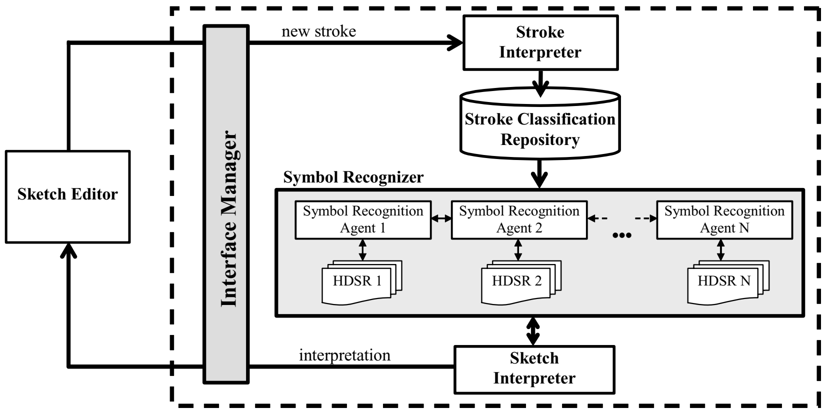

Figure 1. The architecture of the agent-based framework for sketch recognition

identifying key points[1] in order to separate the composite sketch into primitive shapes, such as line and arc segments. In this way, symbols can be drawn with multiple pen strokes, and a single pen stroke can contain multiple symbols. When key points and strokes have been recognized, they are stored into a "Stroke Classification Repository" that can be accessed by all the agents in the system.

Symbol Recognizer. The Symbol Recognizer is composed of a set of SRAs, each one devoted to recognizing a particular symbol of the domain. The main goal of an SRA is to collaborate with other SRAs to apply context knowledge to the symbols they are recognizing, and with the CA that deals with the sketch interpretation activity.

Sketch Interpreter. When one symbol is recognized by one SRA, the SRA sends a message to the CA, included in the Sketch Interpreter, with information about the recognized symbol and its context. Using this information, the CA may provide the correct sketch interpretation to the Interface Manager.

In the following sections we discuss the Symbol Recognizer and the Sketch Interpreter in detail.

3.1 Symbol recognizer

The symbols of the domain language are identified by applying suitable HDSRs to the interpretations produced by the Stroke Interpreter. The execution of these HDSRs is coordinated by SRAs. As already observed, as long as there is one SRA that is able to manage the execution of a set of HDSRs, and to convert data from the stroke repository into a format that the HDSRs can accept, and the output of the HDSRs into a format that the other agents can understand, the actual implementation of the HDSRs and the approach to recognition that they adopt, do not matter. .In Section 4, we exemplify the potential of our approach by outlining how the HDSRs proposed in [10] and [17] could be integrated into our framework. However, many other HDSRs, such as [4, 12, 15, 23], could undergo a similar process.

The main goal of an SRA is to recognize domain symbol instances (as an example the symbols *Actor*, *Communicate*, *UseCase*, *Include*, *Generalize*, and so on, in UML use case diagrams [22]) by managing the execution of a set of HDSRs for a given domain symbol, and by collaborating with other SRAs to obtain contextual feedback. The life cycle of each SRA is characterized by four phases: 1) check the Stroke Classification Repository for new interesting primitive shapes; 2) try to recognize a symbol using the new strokes found during the first step, suitably converted into a format that the underlying HDSRs can accept; 3) collaborate with other SRAs to obtain feedback on the recognition; and 4) interact with the CA.

Checking the repository. When a new stroke becomes available in the repository, each SRA decides whether the stroke may be interesting for recognizing its domain symbol or not.

Recognizing a symbol. We assume that each HDSR, when fed with the proper input that the SRA grabs from the repository and converts into a suitable format, is able to produce an output that contains both the information on the strokes that compose the recognized symbol, and an accuracy that rates the precision of the symbol that has been recognized. This output is associated by the SRA to the symbol,

[1] A key point is a point that contains the most characterizing geometric features of a sketch. For example, a high curvature point, a tangency point, a corner point and an inflexion point.

together with other attributes such as the coordinates of its centre or the minimum enclosing rectangle.

Collaborating with other SRAs. When a symbol has been recognized, the SRA starts the collaboration process to obtain contextual information for the recognized symbol. The collaboration consists of sending a feedback request message containing information about the recognized symbol to all the SRAs that recognize *related* symbols and that are known "a priori" by each SRA.

Two domain symbols are *related* if the domain language defines a relation between them. When an SRA receives a feedback request, it checks its set of recognized symbols to give an answer. If it finds a symbol that satisfies the language relationship with the symbol in the feedback request, it sends a positive response to the requester, otherwise, it sends a negative response. As an example, in UML use case diagrams the *Use Case* symbol is related to *Participate*, *Include* and *Extend* symbols.

Interacting with the Coordinator Agent. When a symbol is recognized and the collaboration phase terminates, the SRA communicates to the CA the information about the new recognized symbol including the set of strokes that form the symbol, its accuracy value, and the positive feedback collected.

3.2 Sketch interpreter

The interpretation of the sketch is demanded to the CA that incrementally analyzes the information received from SRAs and solves conflicts that might arise. When all the conflicts have been solved, the CA proposes the sketch interpretation to the user, interacting with the Interface Manager. The CA looks for conflicts by checking if there are symbols that share one or more strokes. Conflicts may take place either because a stroke is classified as two different shapes (for example, as a line and as an arc) due to the sketch inaccuracy, or because the same stroke, although correctly classified, is used by two SRAs to recognize two different symbols.

In order to support the incremental resolution of conflicts, the CA uses a graph structure to efficiently represent both the information produced by the SRAs, and that obtained during the resolution of the conflicts. In particular, the nodes of the graph correspond to the symbol interpretations provided by the SRAs, whereas the edges can be of two types. The *conflict edges* link conflicting symbols and are labeled with the difference between the accuracy associated to the symbols in absolute value, whereas the *feedback edges* link symbols that have produced a positive feedback during their recognition. The conflict between two symbols is solved in favor of the one having the following higher *truthful* value:

$$tr = w_1\, acc + w_2 (\frac{\#rn}{\#n})$$

where *acc* is the accuracy value of the symbol, *#n* is the total number of nodes, *#rn* is the number of nodes without conflicts (*unambiguous symbols*) reachable by following a feedback edge from the symbol, and w_1 and w_2 are values between 0 and 1 that depend on the domain language. In particular, for languages where symbols in the diagrams are involved in many relations with other symbols, w_2 must be greater than w_1, in order to weight the existence of feedback more than the accuracy of the symbol. Vice versa, for languages with few relations between symbols in diagrams, it is more important to consider the accuracy associated to the symbol, and thus w_1 must be greater than w_2. Unambiguous symbols are used to solve conflicts because they represent stable and not conflicting elements in the current sketch interpretation.

Conflicts are solved starting from:

1. Those that involve one symbol with feedback from unambiguous symbol(s) (*unambiguous feedback*) and one symbol without unambiguous feedback.
2. Those that involve symbols with higher difference between the number of unambiguous feedback.
3. Those that involve symbols with higher difference of accuracy value.

This criterion helps in solving the "easiest" conflicts first, in order to obtain new unambiguous symbols that can be used to solve other conflicts.

When a conflict is solved, the graph is updated. When a new symbol is communicated to the CA, a new node is added to the graph together with the corresponding conflict and feedback edges. The conflict resolution is applied to that portion of the graph reachable from the new node without involving those parts of the diagram that are not related with the added symbol.

In order to reduce the number of active HDSRs, the CA selects and communicates to the SRAs the ones that can be pruned. Many heuristics can be chosen: for example, pruning could be applied to HDSRs that have recognized symbols without feedback, and are involved in conflicts with symbols having feedback, or to HDSRs recognizing symbols whose constituent strokes all belong to another symbol with more positive feedback, and so on. The choice of the heuristics to be applied also depends from the diagrammatic language.

4. Integrating heterogeneous hand-drawn symbol recognizers

In this section, we discuss how our agent-based framework might be used to integrate two different HDSRs and to apply our context-driven cooperative strategy for disambiguating the sketch interpretation process. The two HDSRs that we use to exemplify our approach take inspiration from LADDER [17] and from Sketch Grammars [10].

LADDER. In LADDER, the symbol recognition is performed using the rule-based system Jess (http://herzberg.ca.sandia.gov/jess/). In particular, for each symbol of the domain, a Jess rule is automatically generated from a LADDER structural shape description. A Jess rule-based system always searches for combinations of facts that can satisfy a rule. In the symbol recognition domain, searching for a combination of facts that satisfy a rule means searching for a combination of stroke classifications that represents a domain symbol.

In our example of use case diagrams recognition, the Generalize SRA might demand the actual symbol recognition to a HDSR constructed from a LADDER specification. The "Generalize" symbol of UML use case diagrams is represented by an arrow with a triangle head. The Jess rule used by the "Generalize HDSR" would be:

```
(defrule GeneralizeCheck
;; get four lines
?f0 <- (Subshapes Line ?shaft   \$?shaft_list)
?f1 <- (Subshapes Line ?head1  \$?head1_list)
?f2 <- (Subshapes Line ?head2  \$?head2_list)
?f3 <- (Subshapes Line ?head3  \$?head3_list)
;; make sure lines are unique
(test (uniquefields \$?shaft_list \$?head1_list))
(test (uniquefields \$?shaft_list \$?head2_list))
(test (uniquefields \$?shaft_list \$?head3_list))
(test (uniquefields \$?head1_list \$?head2_list))
(test (uniquefields \$?head1_list \$?head3_list))
(test (uniquefields \$?head2_list \$?head3_list))
;; get accessible components of each line
(Line ?shaft ?shaft_p1 ?shaft_p2 ?shaft_length ?shaft_acc)
(Line ?head1 ?head1_p1 ?head1_p2 ?head1_midpoint ?head1_length ?head1_acc)
(Line ?head2 ?head2_p1 ?head2_p2 ?head2_midpoint ?head2_length ?head2_acc)
(Line ?head3 ?head3_p1 ?head3_p2 ?head3_midpoint ?head3_length ?head3_acc)
;;test constraints
(test (perpendicular ?shaft ?head1))
(test (coincident ?shaft ?head1_midpoint))
(test (coincident ?head1_p2 ?head2_p1))
(test (coincident ?head2_p2 ?head3_p1))
(test (coincident ?head3_p2 ?head1_p1))
(test (equalLength ?head1 ?head2))
(test (acuteMeet ?head1 ?head2))
(test (acuteMeet ?head2 ?head3))
(test (acuteMeet ?head3 ?head1))
=> ;; Generalize symbol found
;; add symbol to recognized symbol
   (computeaccuracy ?shaft_acc ?head1_acc ?head2_acc ?head3_acc ?acc)
   (addshape Generalize ?shaft ?head1 ?head2 ?head3 ?acc))
```

The above rule, like all Jess rules, is composed of two parts. The part at the left of the "=>" symbol contains the name of the rule ("GeneralizeCheck") and the preconditions that enable the rule to fire namely: getting four lines, making sure that the four lines are unique, getting the components of each line, and finally checking that the lines' components meet the topological and geometric constraints that allow an arrow to be composed with them. The part at the right of the "=>" symbol, defines what to do when the precondition is met; in this case, the symbol, together with its constituent parts and its accuracy, computed from the accuracy values produced by the primitive shape recognizer, is added to the set of symbols recognized by the Generalize SRA by calling the "addshape" function.

Thus, when in the Stroke Classification Repository, there are four lines that respect the precondition of the rule, the rule is fired and a Generalize symbol is recognized.

Sketch Grammars. *Sketch Grammars* (SkGs) represent a direct extension of context-free string grammars, where more general relations other than concatenation are allowed [10]. The symbol recognizers automatically generated from SkGs try to cluster stroke interpretations into symbols of the domain language. The parsing technique extends the approaches proposed in [9]: the parsers scan the input in an incremental and non-sequential way, driven by the spatial relations specified by the grammar productions.

An SkG *G* can be seen as a context-free string attributed grammar where the productions have the following format:

$$A_{\Gamma} \rightarrow x_1\, \mathbf{R_1}\, x_2\, \mathbf{R_2} \ldots x_{m-1}\, \mathbf{R_{m-1}}\, x_m,\ Act$$

A is a nonterminal symbol, each x_j is a terminal or nonterminal symbol, and each $\mathbf{R_j}$ is a sequence of spatial and/or temporal relations [10]. *Act* specifies the actions that have to be executed when the production is reduced during the parsing process. These may include a set of rules used to synthesize the values of the attributes of A from those of $x_1, x_2, \ldots, x_m$. Actions are enclosed into the brackets { }. Γ is used to dynamically insert new terminal shapes in the input during the parsing process, enhancing the expressive power of the formalism.

To go on with our example UML use case diagrams recognition, the Actor SRA might manage an "Actor HDSR" implemented using SkG. This HDSR would use the following production to recognize the *Actor* symbol:

$$
\begin{aligned}
\text{Actor} \rightarrow\ & \text{Ellipse} < \mathit{joint}_{1\_1}(t_1) > \\
& \text{Line}_1 < \mathit{near}(t_2),\ \mathit{near}^1(t_3) > \\
& \text{Line}_2 < \mathit{joint}_{2\_1}(t_4),\ \mathit{near}^1(t_5),\ \mathit{near}^2(t_6) > \\
& \text{Line}_3 < \mathit{joint}_{2\_1}{}^2(t_7),\ \mathit{rotate}^2(135,t_8) > \\
& \text{Line}_4 < \mathit{joint}_{2\_1}{}^3(t_9),\ \mathit{rotate}^3(135,t_8) > \text{Line}_5, \\
\{\ & \text{Actor.attach}(1) = \text{Ellipse.attach}(1) \cup \text{Line}_1\text{.attach}(1); \\
& \text{Actor.accuracy} = \text{ComputeAccuracy}();\}
\end{aligned}
$$

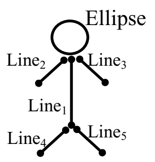

Figure 2. The Actor Symbol

The *Actor* symbol is composed of an ellipse and five lines, as shown in Fig. 2 (the attributes are represented with bullets). The non-terminals *Ellipse* and *Line* cluster the single stroke arcs that form an ellipse and the parallel single stroke lines, respectively. The attribute 1 of *Ellipse*, which represents its borderline, is jointed to the attributes 1 of $Line_1$, $Line_2$, and $Line_3$. The latter are rotated with respect to the former of 45 and -45 degrees, respectively. The values $t_1,\ldots,t_9$ specify the error margin in the satisfaction of the relations. Finally, the attribute 1 of *Actor* is calculated from the values of the attributes of *Ellipse* and $Line_1$, and the accuracy of *Actor* is computed by the *ComputeAccuracy* function, which combines the accuracy of the strokes forming the sketch and of their spatial relations.

Putting all together. Using our framework, the HDSRs generated from LADDER and SkG specifications might be seamlessly integrated, thanks to the definition of suitable SRAs providing a sort of middle layer between the CA and the actual recognition process.

Fig. 3 shows the recognition process of a use case diagram, where the numbers associated to the strokes in the left-top side of the figure denote the temporal sequence of the drawing process. For each symbol to be recognized (*Actor, Generalize, Participate, Use Case, Extend, Include*) an appropriate SRA and an appropriate HDSR are included in the framework. In our example, the Actor HDSR exploits SkG and the Generalize HDSR is based on LADDER, and we do not put constraints on the other HDSRs. When we move from the hand-drawn symbol recognition level, to the recognizer agent level, the underlying recognizing techniques become irrelevant for the communication and coordination purposes of the agents.

While the user draws, the sketch classifications produced by the stroke interpreter are stored in the repository. Each SRA transforms stroke classifications that are interesting to it, into suitable representations that can be used by the underlying HDSR.

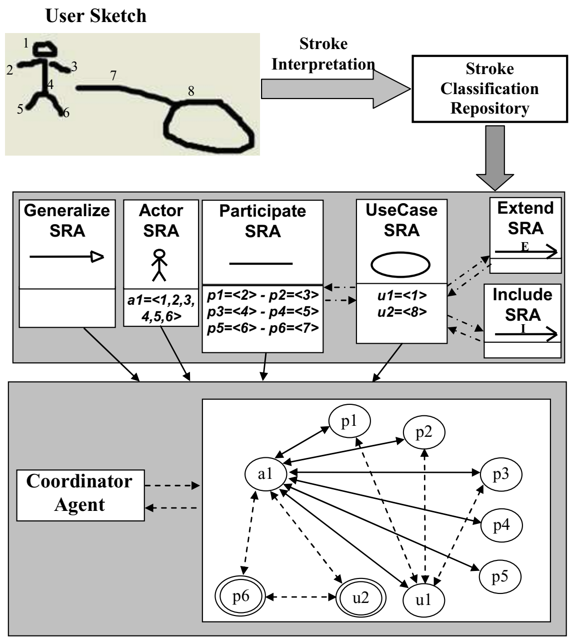

Figure 3. The Recognition Process

When the strokes from 1 to 6 in Fig. 3 are drawn, the HDSR associated to Actor SRA uses the production rule illustrated before to recognize the *Actor* symbol *a1*. Moreover, stroke 1 is also recognized as the *Use Case* symbol *u1* by the HDSR associated to *Use Case SRA,* whereas the line strokes from 2 to 6 are recognized as the *Participate* symbols *p1*, *p2*, *p3*, *p4*, and *p5* by the HDSR associated to *Participate SRA.* Finally, strokes 7 and 8 are correctly recognized as the *Participate* symbol *p6* and *UseCase* symbol *u2*, respectively.

As described in section 3.1, when an SRA recognizes a symbol it starts to collaborate with SRAs recognizing related symbols for obtaining contextual information. In use case diagrams, the Use Case symbol is related to Participate, Include and Extend. Thus, when the Use Case SRA recognizes *u2*, it sends a feedback request to Participate SRA, Extend SRA, and Include SRA. The first replies with a positive response, since *u2* is correctly related to *p6*, while the others reply with a negative response.

When the collaboration phase terminates, the SRAs send to the CA the recognized symbols with their attributes and collected positive feedbacks. At the bottom of Fig. 3 the graph constructed by the CA using the symbols communicated by SRAs is shown. In the graph, conflict edges and feedback edges are visualized with continuous arrows and dashed arrows, respectively. Symbol *a1* is in conflict with several symbols (*p1*, *p2*, *p3*, *p4, p5, u1*). The first conflict that is solved is the one between *a1* and *p1*. Indeed, *a1* collected two unambiguous feedback from *p6* and *u2*, while *p1* did not receive unambiguous feedback (the one from *u1* is not unambiguous). Supposing that *a1* has a greater truthful value than *p1*, *a1* wins the conflict. The conflict resolution goes on and since *a1* wins all its conflicts, it becomes an unambiguous symbol providing unambiguous feedback.

5. Related work

In the last two decades several approaches have been proposed for the recognition of freehand drawings. The novelty of our work consists in the exploitation of intelligent agents for the integration and coordination of heterogeneous hand-drawn symbol recognizers.

If we consider the agent technology, that mainly characterizes our approach, we find that very few approaches are based on it. One of the oldest systems we are aware of is QuickSet, a suite of agents for multimodal human-computer communication [8]. A very similar, but more recent, agent-based multimodal system is Demo, described in [14]. In [1], Achten and Jessurun discuss how graphic unit recognition in drawings can take place using a multi-agent systems approach, where singular agents may specialize in graphic unit-recognition, and multi-agent systems can address problems of ambiguity through negotiation mechanisms. In [21], Mackenzie and Alechina propose an agent-based technique for the classification and understanding of child-like sketches of animals, using a live pen-based input device. Finally, in [18] Juchmes and Leclercq describe EsQUIsE, an interactive tool for free-hand sketches to support early architectural design.

When we compare our proposal with those using the agent technology, we find that the main differences lie in the intended usage domain of the system, which is very specific for all the implemented systems, and in the technique exploited for recognizing symbols from stroke classifications, that is established once and for all by the other approaches.

As far as the first difference is concerned, the only general-purpose view is provided by Achten and Jessurun that, however, do not propose a concrete MAS architecture, but just analyze the feasibility of adopting multi-agent techniques to sketch recognition.

Regarding the construction of symbol recognizers, many approaches have been proposed [4, 10, 12, 15, 17, 23]. We believe that each of these proposals can be integrated into our framework in spite of the fact that they differ one from another under several aspects, ranging from the identification of the shape of the symbols to the approach used to construct them. For instance, the Rubine recognition engine is a trainable recognizer for single stroke gestures [23]. Gestures are represented by global features and are classified according to a linear function of the features. However, the recognizer is applicable to single-stroke sketches and is sensitive to the drawing direction and orientation. In [4] Apte *et al.* developed a hard-coded recognizer that examines the geometric properties of the convex hull of a symbol. The recognizer also makes use of special geometric properties of particular shapes. CALI is a system for recognizing multi-stroke geometric shapes based on a naïve Bayesian classifier [12]. Fuzzy logic is also employed in their graphics recognition approach such that their recognition approach is orientation independent. Nevertheless, the filters applied during the recognition are ineffective on ambiguous shapes such as pentagon and hexagon. In [15] Kara and Stahovich have developed a symbol recognizer that is capable of learning new definitions from single prototype examples. Moreover, since it is based on a down-sampled bitmap representation, it is particularly useful for drawings with heavy over-stroking and erasing.

6. Conclusions and future work

In this paper we have presented an agent-based framework for interpreting symbols in diagrammatic sketches in a context-driven fashion, exploiting heterogeneous techniques for the recognition of each single symbol of the language. The recognition process is supported by intelligent agents (SRAs) that manage the activity of hand-drawn symbol recognizers, and coordinate themselves in order to provide efficient and precise interpretations of the sketch to the user. In a certain sense, SRAs can be seen as *mediators* that implement the middle layer for integrating information provided by different data sources (the HDSRs). The approach to information mediation based on intelligent agents has a long tradition [5, 26]. In our approach we apply the ideas behind mediation to a new research field.

Currently, the main limitation of our framework is that it defines a high-level architecture where heterogeneous components can be suitably integrated thanks to intelligent agents, but neither methodological guidelines for performing this integration, nor software application for supporting them, has been developed yet. The system exemplified in Section 4 is, in fact, under implementation; we have described how it will look like and what it will do, but we cannot provide experimental results since a complete working prototype is still missing.

Our short term future work is thus to complete the implementation of our case study in order to provide a set of already developed SRAs (that for integrating LADDER, that for integrating Sketch Grammars, etc.) from which the developer can choose when building his/her own system.

In the medium term, on the one hand we will define a set of methodological guidelines for integrating heterogeneous HDSRs into the framework, and on the other hand we will explore the advantages of integrating a parser into the coordinator agent for analyzing the syntax of the diagrammatic language. This will allow us to improve the accuracy in computing feedback information and to reduce the number of symbol recognition processes.

References

[1] H.H. Achten and A.J. Jessurun, "An Agent Framework for Recognition of Graphic Units in Drawings", in *Proc. of eCAADe'02*, 2002, pp. 246-253.

[2] M. Amer, A. Karmouch, T. Gray, S. Mankovski, "An Agent Model for Resolution of Feature Conflicts in Telephony", *J. Networks Syst. Manage*, 8(3), 2000, pp. 419-437.

[3] C. Alvarado, and R. Davis, "Dynamically Constructed Bayes Nets for Multi-Domain Sketch Understanding", in *Proc. of IJCAI'05*, 2005, pp. 1407-1412.

[4] A. Apte, V. Vo, and T.D. Kimura, "Recognizing multistroke geometric shapes: An experimental evaluation", in *Proc. UIST 93*, 1993, pp. 121-128.

[5] S. Bergamaschi, "Extraction of Informations from highly Heterogeneous Sources of Textual Data", in *Proc. of CIA'97*, LNCS, 1997, pp. 42-63

[6] D. Blostein and L. Haken, "Using Diagram Generation Software to Improve Diagram Recognition: A Case Study of Music Notation", *IEEE Transactions on PAMI*, 21(11), 1999, pp. 1121-1136.

[7] J. Chu-Carroll and S. Carberry, "Communication for Conflict Resolution in Multi-Agent Collaborative Planning", in *Proc. of ICMAS'95*, pp. 49-56.

[8] P. R. Cohen, M. Johnston, D. McGee, I. Smith, J. Pittman, L. Chen, and J. Clow, "Multimodal interaction for distributed interactive simulation", in *Proc. of IAAI'97*, pp. 978-985.

[9] G. Costagliola, V. Deufemia, G. Polese, M. Risi, "A Parsing Technique for Sketch Recognition Systems", in *Proc. of IEEE VL/HCC'04*, pp. 19-26.

[10] G. Costagliola, V. Deufemia, M. Risi, "Sketch Grammars: A Formalism for Describing and Recognizing Diagrammatic Sketch Languages", in *Proc. of ICDAR'05*, IEEE Press, pp. 1226-1230.

[11] J.L.T. da Silva and V.L. Strube de Lima, "Lexical Categorical Disambiguation using a Multi-Agent Systems Architecture", in *Proc. of ICMAS'98*, pp. 417-418.

[12] M.J. Fonseca, C. Pimentel, and J.A. Jorge, "CALI – An Online Scribble Recognizer for Calligraphic Interfaces", in *Proc. AAAI Symp. Sketch Understanding*, 2002, pp. 51-58.

[13] M.D. Gross, "The Electronic Cocktail Napkin – A Computational Environment for Working with Design Diagrams", *Design Studies*, 17(1), pp. 53-69, 1996.

[14] E. Kaiser, D. Demirdjian, A. Gruenstein, X. Li, J. Niekrasz, M. Wesson, and S. Kumar, "Demo: A Multimodal Learning Interface for Sketch, Speak and Point Creation of a Schedule Chart", in *Proc. of ICMI'04*, 2004, pp. 329-330.

[15] L.B. Kara and T.F. Stahovich, "An Image-based, Trainable Symbol Recognizer for Hand-drawn Sketches", *Computers & Graphics*, 29(4), 2005, pp. 501-517.

[16] T. Hammond and R. Davis, "Tahuti: A Geometrical Sketch Recognition System for UML Class Diagrams", in *Proc. of the AAAI Symp. on Sketch Understanding*, 2002, pp. 51-58.

[17] T. Hammond and R. Davis, "LADDER, A Sketching Language for User Interface Developers", *Computers & Graphics*, 29(4), 2005, pp. 518-532.

[18] R. Juchmes and P. Leclercq, "A Multi-Agent System for the Interpretation of Architectural Sketches", in *Proc. of 2004 Eurographics Workshop on Sketch-Based Interfaces and Modeling*, Grenoble, France, pp. 53-61.

[19] N.R. Jennings, K. P. Sycara, and M. Wooldridge, "A Roadmap of Agent Research and Development", *Journal of Autonomous Agents and Multi-Agent Systems*, 1(1), 1998, pp 7-36.

[20] M. Luck, P. McBurney, O. Shehory, S. Willmott, and the AgentLink Community, "Agent Technology: Computing as Interaction – A Roadmap for Agent-Based Computing", *AgentLink III*, 2005.

[21] G. Mackenzie and N. Alechina, "Classifying Sketches of animals using an Agent-Based System", *Proc. of CAIP 2003*, LNCS 2756, pp. 521-529.

[22] Object Management Group. *UML Specification version 2.0*. http://www.omg.org/technology/documents/formal/uml.htm

[23] D. Rubine, "Specifying Gestures by Example", *Computer Graphics*, 25(4), 1991, pp. 329-337.

[24] T.M. Sezgin, "Feature point detection and curve approximation for early processing in sketch recognition", *Master's thesis*, Massachusetts Institute of Technology, June 2001.

[25] T.F. Stahovich, R. Davis, and H. Shrobe, "Generating Multiple New Designs from a Sketch", *Artificial Intelligence*, 104(1-2), 1998, pp. 211-264.

[26] V. S. Subrahmanian, S-S. Chen, J. A. Hendler, R. Hull and V. Tannen, "Smart Mediators and Intelligent Agents (Panel)", in *Proc. CIKM* 1996, page 343.

[27] M. Wooldridge and N. R. Jennings. "Intelligent agents: Theory and practice", *The Knowledge Engineering Review*, 10(2), 1995, pp. 115-152.