Symbolic Agent Negotiation for Semantic Web Service Exploitation

Peep Küngas[1], Jinghai Rao[1], and Mihhail Matskin[2]

[1] Norwegian University of Science and Technology
Department of Computer and Information Science
Trondheim, Norway
{peep,jinghai}@idi.ntnu.no
[2] Royal Institute of Technology
Department of Microelectronics and Information Technology
Kista, Sweden
misha@imit.kth.se

Abstract. This paper presents an architecture and a methodology for agent-based Web service discovery and composition. We assume that Web services are described with declarative specifications like DAML-S. Based on the declarative information about services, symbolic reasoning can be applied while searching for or composing automatically new services.
We propose that symbolic agent negotiation could be used for dynamic Web service discovery and composition. Symbolic negotiation, as we consider it here, is a mixture of distributed planning and information exchange. Therefore, by using symbolic negotiation for automated service composition, we support information collection and integration during service composition. The latter aspect has been largely neglected in automated service composition until now.

1 Introduction

Several initiatives in Web services provide platforms and languages that should allow easy integration of heterogeneous systems. In particular, such standards as UDDI, WSDL, SOAP and a part of DAML-S ontology define standard ways for service discovery, description and invocation. Some other initiatives such as BPEL4WS and DAML-S ServiceModel, are focused on representing service composition where flow of a process and bindings between services are known a priori.

The ability to efficiently select and integrate inter-organisational and heterogeneous Web services at runtime is an important requirement to the Web service provision. In particular, if no single Web service can satisfy the functionality required by a user, there should be a possibility to combine existing services together in order to fulfill the request. A challenge is that Web services can be created and updated on the fly and it may be beyond human capabilities to analyze the required services and compose them manually. As a result,

Q. Li, G. Wang, and L. Feng (Eds.): WAIM 2004, LNCS 3129, pp. 458–467, 2004.

the service composition becomes a complex problem, which definitely cannot be solved without some degree of automation.

Several articles address automatic composition of Web services [10, 13, 14, 16]. However, they all require existence of a central directory of Web service specifications, which contrast largely to the dynamic nature of the Web. In the Web the set of available services changes rapidly – new services are created, old ones are modified or removed. Keeping track of all these changes is a huge burden for a centralised directory. Some essential issues in decentralised Web service provision have been addressed by Papazoglou et al [11].

Another disadvantage of a centralised approach is that it only allows service requesters to locate services, while service providers lack an ability to attract potential customers. The agent-based architecture we propose here gives service providers a more proactive role in the service composition process. Our service provision architecture is based on the multi-agent system AGORA [8], which provides an infrastructure, where service providers and requesters can meet with each-other.

We propose that symbolic agent negotiation could be used as a mechanism for discovering available Web services and composing new ones automatically. If no service, satisfying user's requirements, is found, symbolic negotiation between agents is initiated and a new composite Web service is constructed dynamically. We take advantage of symbolic negotiation framework described by Küngas and Matskin [4, 5], where linear logic [3] (LL) is applied to agent negotiation.

The work presented in this article is closely related to framework in [12], where LL theorem proving is applied for automated Web service synthesis in a centralised manner. In this article we go beyond the centralised approach and propose how to take advantage of intelligent agent technologies for distributed Web service composition. We assume that service descriptions have been already translated into LL formalism. Therefore we do not describe mapping from DAML-S to LL formulae.

The rest of the paper is organised as follows. In Section 2 we give a general description of the system architecture. Section 3 formalises symbolic negotiation for agent systems. Section 4 demonstrates usage of symbolic negotiation for Web service composition. Section 5 reviews the related work and Section 6 concludes the paper.

2 The System Architecture

The AGORA multi-agent environment [8] was developed with the intention to support cooperative work between agents. The system consists of 2 types of components (nodes) – agents and agoras. Agoras are cooperative nodes which facilitate agent communication, coordination and negotiation. Moreover, agoras encapsulate a method for (partial) service composition. An agora node contains *default agents* and *registered agents*. In our scenario, the default agents are Agora Manager and Negotiator. Agora Manager implements general agora functions,

such as service matchmaking and agent/service registration, while Negotiator applies symbolic negotiation.

Service matchmaking basically involves finding an atomic service, which satisfies agent requirements for a service. Negotiator applies symbolic negotiation for composing new services automatically. Negotiation is applied, if Agora Manager failed to find an atomic service satisfying agents' requirements.

Service provider agents register their services at specific agoras according to their service domains. For example agents providing services for selling, buying and managing hardware register themselves and available services at agoras devoted to hardware. Specific agoras may represent also coalitions of service providers.

Service requester agents, however, register requests for services at the central agora. Then the central agora negotiates with specific agoras to find services satisfying requester agents' requirements. The central agora also mediates information exchange between different requester agents. Moreover, the central agora might also form requester agent coalitions, if it is needed for certain services. It may happen for instance that a (composite) service requires input from more that one service requester agent. Our system architecture is depicted in Fig. 1.

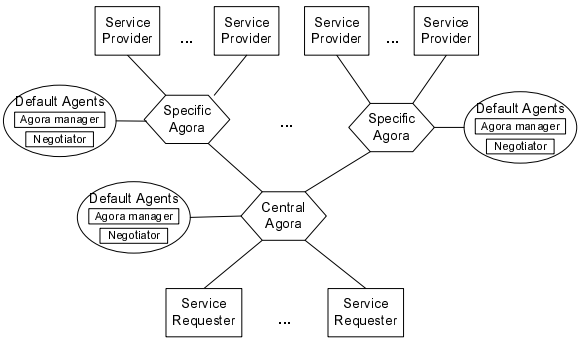

Fig. 1. The system architecture.

A specific service composition architecture can be specified through refinement and instantiation of the generic architecture described above. An instance of the generic architecture is elaborated in Section 4 and depicted in Fig. 3.

Fig. 2 presents the proposed interaction protocol in Agent UML notation. This protocol uses the FIPA (The Foundation for Intelligent Physical Agents) reserved communicative acts, allowing thus interoperability with other FIPA compliant agent systems. The agent interaction process is summarised in the following.

The central agora is a search control point for requester agents' queries. Specific agoras are cooperative nodes gathering agents who provide the same sort of services. Specific agoras register themselves to the central agora. After receiving the registration request from a service provider agent, the central agora

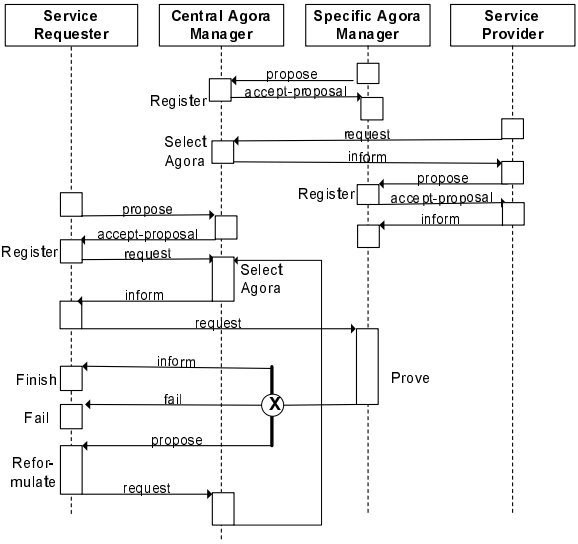

Fig. 2. The interaction protocol.

locates a specific agora where the service provider could register itself. In our case, the service providers registered to the same specific agora provide services in the same domain.

Service requester agents register themselves to the central agora. Additionally they publish their requirements in a declarative language. After that the central agora tries to locate an atomic service satisfying the requirements. However, if no suitable atomic service is found, symbolic negotiation is initiated. During symbolic negotiation the central agora contacts specific agoras to receive a composite service for satisfying the particular requirements.

It might happen that services from different agoras are needed for a composition. In that case the central agora receives a partial composition from one specific agora and forwards it to another specific agora for further composition. The process is called symbolic negotiation – the central agora negotiates with specific agoras to compose a solution for a service requester agent. Finally a composite service is constructed and returned to the requester agent, who initiated the symbolic negotiation.

3 Symbolic Negotiation

In this section we present formal foundations of symbolic agent negotiation as it was proposed in [4, 5]. Symbolic negotiation is generally defined as an interactive process involving partial deduction (PD) and agent communication. Throughout this paper we consider PD only for propositional LL, since Web services in DAML-S can be represented in a propositional LL fragment. However, the extension in [5] allows usage of first-order LL as well for symbolic negotiation.

For representing symbolic negotiation we consider !Horn fragment of LL (HLL) consisting of multiplicative conjunction ($\otimes$), linear implication ($\multimap$) and "of course" operator (!). From symbolic negotiation point of view the logical expression $A \vdash C$ means that an agent can provide A and requires C. ! represents unbounded access to resources. Thus $!A$ means that resource A can be used as many times as needed. To increase the expressiveness of formulae, we are using in the following abbreviation $a^n = \underbrace{a \otimes \ldots \otimes a}_{n}$, for $n \geq 0$, with the degenerate case $a^0 = 1$.

An agent is presented with the following LL sequent $\Gamma; S \vdash G$, where Γ is a set of extralogical LL axioms representing agent's capabilities (services), S is the initial state and G is the goal state of an agent. Both S and G are multiplicative conjunctions of literals. Every element of Γ has the form $\vdash I \multimap O$, where I and O are formulae in conjunctive normal form which are, respectively, consumed and generated when a particular capability is applied. It has to be mentioned that a capability can be applied only, if conjuncts in I form a subset of conjuncts in S.

Messages are defined with the tuple (id_{req}, S, R, O), where id_{req}, S, R and O denote respectively message identifier, sender, receiver and offer. Message identifier is needed to keep track of different negotiations. Sender and receiver are identifiers of participating agents and offer is represented with a LL sequent in form $A \vdash B$, where A and B are multiplicative conjunctions of literals.

In [4] PD steps $\mathcal{R}_b(L_i)$ and $\mathcal{R}_f(L_i)$ were defined for back- and forward chaining:

$$\frac{S \vdash B \otimes C}{S \vdash A \otimes C}\ \mathcal{R}_b(L_i) \qquad \frac{A \otimes C \vdash G}{B \otimes C \vdash G}\ \mathcal{R}_f(L_i)$$

L_i in the inference figures is a labelling of a particular LL axiom representing an agent's capability (computability clause in PD) in the form $\vdash B \multimap_{L_i} A$. $\mathcal{R}_f(L_i)$ and $\mathcal{R}_b(L_i)$ apply clause L_i to move the initial state towards the goal state or the other way around. A, B and C are formulae in HLL.

4 An Example

In this section we demonstrate the usage of symbolic negotiation in distributed service composition. Before we start the example, we have to emphasize two assumptions. First, we assume the users use standard Web service languages, e.g. DAML-S to specify the Web services. The logical formulas used in the example can be translated from DAML-S by the method introduced in [12]. Second, the Web services here include both information-gathering services and world-altering services. DAML-S has capability to distinguish the two kinds of services [10]. In general, the "input" and "output" properties provide information about dataflow, while the "precondition" and "effect" properties tell what the Web service actually does. Hence the Web service in the example not only can present the information flow, but also can indicate the exchange of the goods, such as CD player and books.

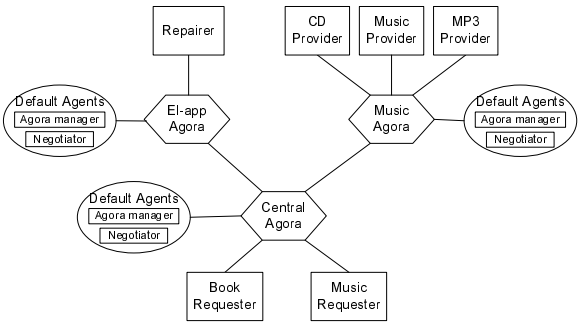

Fig. 3. The example architecture.

The particular system architecture and its components are depicted in Fig. 3. In our scenario we have two requester agents – $\mathcal{R}_1$ and $\mathcal{R}_2$. The goal of $\mathcal{R}_1$ is to listen a music (*Music*). Initially $\mathcal{R}_1$ has a book (*Book*), a broken CD player (*BrokenCDPlayer*) and 5 dollars ($Dollar^5$). Goals, resources and capabilities of $\mathcal{R}_1$ are described in LL with the following formulae.

$$G_{\mathcal{R}_1} = \{Music\}, \quad S_{\mathcal{R}_1} = \{Book \otimes BrokenCDPlayer \otimes Dollar^5\}, \quad \Gamma_{\mathcal{R}_1} = \emptyset.$$

Another query agent $\mathcal{R}_2$ is looking for a book (*Book*) and is in possession of 10 dollars ($Dollar^{10}$). Goals, resources and capabilities of the query agent $\mathcal{R}_2$ are described in LL with the following formulae.

$$G_{\mathcal{R}_2} = \{Book\}, \qquad S_{\mathcal{R}_2} = \{Dollar^{10}\}, \qquad \Gamma_{\mathcal{R}_2} = \emptyset.$$

In addition we have several service provider agents. However, since they published their services through particular agoras and these agoras take care of advertising the services, we do not list here the internal states of the service provider agents. Instead we present the internal states of agoras. Although agoras may have their own resources and goals, which determine their policies, we consider here only a simple case, where agoras take advantage of registered services/capabilities only.

According to the literals service providers can "produce", the providers are aggregated into two agoras – one for music and another for electrical appliances. In the Music Agora $\mathcal{M}$, three agents, *MusicProvider*, *CDProvider* and *MP3Provider*, can provide services related to music. Services *playCD* and *playMP3* provide respectively knowledge about requirements for playing CD-s and MP3-s. Services *buyMP3* and *buyCD* provide means for ordering particular music media.

$$\Gamma_{\mathcal{M}} = \begin{array}{l} \vdash_{MusicProvider} CD \otimes CDPlayer \multimap_{playCD} Music, \\ \vdash_{MusicProvider} MP3 \otimes MP3Player \multimap_{playMP3} Music, \\ \vdash_{CDProvider} Dollar^5 \multimap_{buyCD} CD, \\ \vdash_{MP3Provider} Dollar^3 \multimap_{buyMP3} MP3. \end{array}$$

The agora $\mathcal{E}$ is an aggregation of the agents who can provide electrical appliances. It only advertises one service *repair* from agent *Repairer*. The service description declares that the agent can repair a CD player by charging 10 dollars.

$$\Gamma_{\mathcal{E}} = \vdash_{Repairer} Dollar^{10} \otimes BrokenCDPlayer \multimap_{repair} CDPlayer.$$

Let us look now how symbolic negotiation is applied for constructing dynamically services, which satisfy users' goals. Initially the query agent $\mathcal{R}_1$ sends out a query to agora $\mathcal{M}$ for finding a service satisfying its requirements:

$$(o_1, \mathcal{R}_1, \mathcal{M}, Book \otimes BrokenCDPlayer \otimes Dollar^5 \vdash Music).$$

The query would be satisfied by a service

$$\vdash Book \otimes BrokenCDPlayer \otimes Dollar^5 \multimap Music.$$

Unfortunately the service requirement is too specific and no matching service is found. However, agora $\mathcal{M}$ modifies the received offer and sends back the following offers:

$$(o_2, \mathcal{M}, \mathcal{R}_1, Book \otimes BrokenCDPlayer \vdash CDPlayer)$$

and

$$(o_3, \mathcal{M}, \mathcal{R}_1, Book \otimes BrokenCDPlayer \otimes Dollar^2 \vdash MP3Player),$$

which were deduced through PD in the following way:

$$\dfrac{\dfrac{\dfrac{Book \otimes BrokenCDPlayer \vdash CDPlayer}{Book \otimes BrokenCDPlayer \otimes Dollar^5 \vdash Dollar^5 \otimes CDPlayer}\ normalise}{Book \otimes BrokenCDPlayer \otimes Dollar^5 \vdash CD \otimes CDPlayer}\ \mathcal{R}_b(buyCD)}{Book \otimes BrokenCDPlayer \otimes Dollar^5 \vdash Music}\ \mathcal{R}_b(playCD)$$

$$\dfrac{\dfrac{\dfrac{Book \otimes BrokenCDPlayer \otimes Dollar^2 \vdash MP3Player}{Book \otimes BrokenCDPlayer \otimes Dollar^5 \vdash Dollar^3 \otimes MP3Player}\ normalise}{Book \otimes BrokenCDPlayer \otimes Dollar^5 \vdash MP3 \otimes MP3Player}\ \mathcal{R}_b(buyMP3)}{Book \otimes BrokenCDPlayer \otimes Dollar^5 \vdash Music}\ \mathcal{R}_b(playMP3)$$

where *normalise* is another LL inference figure, which reduces the number of literals from a particular sequent:

$$\dfrac{\dfrac{\dfrac{}{A \vdash A}\ Id \quad \dfrac{}{B \vdash C}}{B, A \vdash C \otimes A}\ R\otimes}{B \otimes A \vdash C \otimes A}\ L\otimes$$

Agent $\mathcal{R}_1$ chooses the offer o_2 and forwards it to the electrical appliance agora $\mathcal{E}$:

$$(o_4, \mathcal{R}_1, \mathcal{E}, Book \otimes BrokenCDPlayer \vdash CDPlayer).$$

Agora $\mathcal{E}$ modifies the offer further and ends up with the following counteroffer:

$$(o_5, \mathcal{E}, \mathcal{R}_1, Book \vdash Dollar^{10}),$$

which was derived in the following way:

$$\dfrac{\dfrac{Book \vdash Dollar^{10}}{Book \otimes BrokenCDPlayer \vdash Dollar^{10} \otimes BrokenCDPlayer}\ normalise}{Book \otimes BrokenCDPlayer \vdash CDPlayer}\ \mathcal{R}_b(repair)$$

Since no service provider can produce the *Dollar* literal, the message is sent to the central Agora, which has an overview of requester agents' requirements. Fortunately, it turns out that agents $\mathcal{R}_1$ and $\mathcal{R}_2$ can satisfy mutually their requirements such that requester $\mathcal{R}_1$ gets 10 dollars and requester $\mathcal{R}_2$ gets a book. The resulting service composition can be translated to a process description languages like DAML-S process model or BPEL4WS. The exact translation process is described in another paper and is not covered here.

5 Related Work

Gibbins et al [2] demonstrated, through an implementation, the usage of DAML-S Web service descriptions within agents. Their agents embody DAML-S descriptions of Web services and agent communication is managed through FIPA ACL.

Another step towards incorporating Web services into agents is proposed by Ardissono et al [1]. Their main contribution is support for more rigorous Web service execution protocols compared to currently prevalent 2-step protocols (sending input data and then collecting results). We go beyond that approach by allowing a server agent to compose a sequence of action for a service consumer such that a required sequence of service command executions is constructed automatically at server side and the consumer can provide all details once.

In [10] a modification of Golog [6] programming language is used for automatic construction of Web services. It is argued there that Golog provides a natural formalism for automatically composing services on the Semantic Web. Golog has been enriched with some extralogical constructions like **if**, **while**, etc.

Waldinger [15] proposes initial ideas for another deductive approach. The approach is based on automated deduction and program synthesis and has its roots to work presented in [7]. First available services and user requirements are described with a first-order language, related to classical logic, and then constructive proofs are generated with SNARK theorem prover.

In [16] SHOP2 planner is applied for automatic composition of DAML-S services. Other planners for automatic Web service construction include [13, 9]. Thakkar et al. [14] consider dynamic composition of Web services using mediator agent architecture. The mediator takes care of user queries, generates wrappers around information services and constructs a service integration plan.

6 Conclusions and Further Work

In this paper we described an agent architecture for automatic composition of Web services. Agent-specific aspects provide Web service composition with

proactivity, reactivity, social ability and autonomy, while usage of DAML-S, FIPA ACL and application domain specific ontologies provide a standardised medium for Web service deployment. Usage of DAML-S allows publishing semantically enriched specifications of Web services and thus fits well to the Semantic Web initiative, where both Web services and data are labelled with semantic information.

Given that services are represented in DAML-S, they are translated to LL formulae for internal agent reasoning. Given such representation, agents can employ symbolic agent negotiation for composing new Web services according to their requirements. This approach leads to distributed composition of Web services.

We have implemented a symbolic negotiation framework, which exploits an AI planner called RAPS for symbolic reasoning and automated Web service composition. Additionally we have developed a tool for automatically composing Web services whose descriptions are given in DAML-S. The system applies RAPS planner for centralised Web service composition. Further work would concentrate to the evaluation of decentralised Web service composition. Additionally, we would like to extend our method for service composition to exploit also business models, which specify additional relationships between services.

Acknowledgments

This work was partially supported by the Norwegian Research Foundation in the framework of the Information and Communication Technology (IKT-2010) program – the ADIS project. The authors would also thank the anonymous reviewers for their constructive comments.

References

1. L. Ardissono, A. Goy, and G. Petrone. Enabling conversations with web services. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2003, July 14–18, 2003, Melbourne, Victoria, Australia*, pages 819–826. ACM Press, 2003.
2. N. Gibbins, S. Harris, and N. Shadbolt. Agent-based semantic web services. In *Proceedings of the Twelfth International World Wide Web Conference, WWW2003, Budapest, Hungary, May 20–24, 2003*, pages 710–717. ACM Press, 2003.
3. J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.
4. P. Küngas and M. Matskin. Linear logic, partial deduction and cooperative problem solving. In *Proceedings of the First International Workshop on Declarative Agent Languages and Technologies (in conjunction with AAMAS 2003), DALT'2003, Melbourne, Australia, July 15, 2003*. Springer-Verlag, 2004. To appear.
5. P. Küngas and M. Matskin. Symbolic negotiation with linear logic. In *Proceedings of the Fourth International Workshop on Computational Logic in Multi-Agent Systems, CLIMA IV, Fort Lauderdale, FL, USA, January 6-7, 2004*. Springer-Verlag, 2004. To appear.

6. H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl. Golog: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31(1–3):59–83, 1997.
7. Z. Manna and R. J. Waldinger. A deductive approach to program synthesis. *ACM Transactions on Programming Languages and Systems*, 2(1):90–121, 1980.
8. M. Matskin, O. J. Kirkeluten, S. B. Krossnes, and Østein Sæle. Agora: An infrastructure for cooperative work support in multi-agent systems. In T. Wagner and O. F. Rana, editors, *International Workshop on Infrastructure for Multi-Agent Systems, Barcelona, Spain, June 3–7, 2000, Revised Papers*, volume 1887 of *Lecture Notes in Computer Science*, pages 28–40. Springer-Verlag, 2001.
9. D. McDermott. Estimated-regression planning for interaction with web services. In *Proceedings of the 6th International Conference on AI Planning and Scheduling, Toulouse, France, April 23–27, 2002*. AAAI Press, 2002.
10. S. McIlraith and T. C. Son. Adapting Golog for composition of semantic web services. In *Proceedings of the Eighth International Conference on Knowledge Representation and Reasoning (KR2002), Toulouse, France, April 22–25, 2002*, pages 482–493. Morgan Kaufmann, 2002.
11. M. P. Papazoglou, B. J. Krämer, and J. Yang. Leveraging web-services and peer-to-peer networks. In J. Eder and M. Missikoff, editors, *15th International Conference on Advanced Information Systems Engineering, CAiSE 2003, June 16–18, 2003, Klagenfurt, Austria*, volume 2681 of *Lecture Notes in Computer Science*, pages 485–501. Springer-Verlag, 2003.
12. J. Rao, P. Küngas, and M. Matskin. Application of linear logic to web service composition. In *Proceedings of the First International Conference on Web Services (ICWS 2003), Las Vegas, USA, June 23–26, 2003*, pages 3–9. CSREA Press, 2003.
13. M. Sheshagiri, M. desJardins, and T. Finin. A planner for composing services described in DAML-S. In *Proceedings of the AAMAS Workshop on Web Services and Agent-based Engineering*, 2003.
14. S. Thakkar, C. A. Knoblock, J. L. Ambite, and C. Shahabi. Dynamically composing web services from on-line sources. In *Proceeding of 2002 AAAI Workshop on Intelligent Service Integration, Edmonton, Alberta, Canada*, 2002.
15. R. Waldinger. Web agents cooperating deductively. In *Proceedings of FAABS 2000, Greenbelt, MD, USA, April 5–7, 2000*, volume 1871 of *Lecture Notes in Computer Science*, pages 250–262. Springer-Verlag, 2001.
16. D. Wu, B. Parsia, E. Sirin, J. Hendler, and D. Nau. Automating DAML-S web services composition using SHOP2. In *Proceedings of the 2nd International Semantic Web Conference, ISWC 2003, Sanibel Island, Florida, USA, October 20–23, 2003*, 2003.