Abducção de planos e de intenções em diálogos

Paulo Quaresma* pq@fct.unl.pt
José Gabriel Lopes gpl@fct.unl.pt

Centro de Inteligência Artificial, UNINOVA
2825 Monte da Caparica, Portugal
30 de Julho de 1993

Resumo

Uma interacção robusta em Língua Natural implica a capacidade de inferir, em cada momento, as crenças e as intenções de cada agente. Neste artigo é proposto uma ambiente que suporta o reconhecimento de planos e de intenções através de inferências abdutivas sobre as diversas frases do diálogo. As diferentes acções, eventos, estados e conhecimento do mundo são representados por programas em lógica extendida com negação explícita e o processo de inferência abdutiva é modelado pelo sistema proposto por Pereira ([10]) que é baseado na semântica bem fundada com negação explícita (WFSX) e remoção de contradições (CRSX). Será ainda demonstrado como este sistema suporta planeamento abdutivo com o Event Calculus ([4]) e serão apresentados alguns exemplos clássicos ([7, 11]) no domínio de diálogos cooperativos. Finalmente, serão apresentados alguns problemas em aberto e será descrito algum do trabalho a realizar.

1 Introducção

Uma interacção robusta em Língua Natural implica a capacidade, por parte de cada agente, de inferir as crenças, intenções e planos dos outros agentes participantes no diálogo. Nos últimos anos tem havido diversas abordagens para tentar solucionar este problema. Uma das abordagens principais segue o esquema clássico de planeamento desenvolvido no modelo STRIPS ([5]) e NOAH ([13]). Nestes modelos cada plano é definido como uma sequência de acções, sendo cada acção composta por um cabeçalho, pré-condições, restrições, efeitos e sub-acções. A inferência de planos (uma lista de acções) é feita recorrendo a uma biblioteca de planos e acções, aos possíveis objectivos do utilizador e a algumas regras heurísticas. Esta abordagem foi utilizada por, entre outros autores, Litman e Allen ([7, 6]) de modo a inferir planos a partir de actos de fala em diálogos. Uma abordagem diferente foi seguida por Pollack ([11, 1]) que tem como objectivo modelar os planos como estados mentais e em que a sua inferência é feita através da abducção das atitudes subjacentes aos actos de fala.

*Bolseiro da JNICT, n^o BD/1766/IA

Neste artigo iremos seguir uma abordagem mais geral que permitirá utilizar qualquer uma das abordagens referidas. Para tal, recorrer-se-á à programação em lógica e ao processo de raciocínio não-monotónico (nomeadamente raciocínio por defeito e abductivo) como processo básico de inferência. A representação das acções, eventos e conhecimento do mundo, bem como o processo da sua inferência é feito através do Event Calculus e de programas em lógica. Como ambiente de suporte foi utilizado a semântica bem fundada de programas em lógica extendida com a negação explícita (WFSX) e com remoção de contradições (CRSX) a partir do trabalho de Pereira ([10]). Este ambiente define uma semântica formal a partir da qual é possível modelar diversos tipos de raciocínio não-monotónico, nomeadamente raciocínio por defeito, hipotético e abdutivo.

Na secção 2 é feita uma descrição detalhada deste sistema, com um especial ênfase no processo de modelação do raciocínio não-monotónico. Na secção 3 é descrito o processo de planeamento abdutivo com o Event Calculus e na secção 4 é demostrado como, utilizando este ambiente, é possível suportar as abordagens de Litman e de Pollack. Finalmente, na secção 5 são descritos alguns dos problemas em aberto e do trabalho a realizar.

2 Ambiente de Programação em Lógica

Para modelar o tipo de problemas que pretendemos abordar neste artigo (inferência de planos e atitudes em diálogos) é necessário representar acções, eventos, estados e o conhecimento do mundo. Para tal, utilizaram-se programas em lógica extendida com a negação explícita que são definidos por um conjunto de regras e de restrições de integridade com a seguinte forma:

- $H \leftarrow B_1, \ldots, B_n, notC_1, \ldots, notC_m (m \geq 0, n \geq 0)$

onde H, B_1, ..., B_n, C_1, C_m são literais clássicos. Um literal clássico é um átomo A ou a sua negação explícita $\neg A$. *not* representa a negação por falhanço clássica nas linguagens de programação em lógica. Nas regras de integridade, H é o símbolo $\bot$ (contradicção).

O raciocínio por defeito pode ser modelado adicionando ao programa regras de acordo com o seguinte esquema:

- Normalmente $A(X)$ implica $B(X)$

que pode ser descrita da seguinte forma:

1. B(X) $\leftarrow$ A(X), not ab(X)

definindo que, se não for possível provar a anormalidade de X, então B deve ser válido.

Com uma pequena variação também é possível modelar raciocínio hipotético:

- Os quakers podem ser (ou não) pacifistas

que pode ser descrito da seguinte forma:

1. pacifist(X) ← quaker(X), hypqp(X)
2. hypqp(X) ← not ¬ hypqp(X)
3. ¬ hypqp(X) ← not hypqp(X)

e que define que a hipótese dos quakers serem pacifistas é válida se não se conseguir provar (com a negação por falhanço) o contrário (e vice-versa).

O raciocínio abductivo é modelado através das seguintes regras:

- F pode ou não ser verdadeiro

que pode ser descrito da seguinte forma:

1. $F \leftarrow not \neg F$
2. $\neg F \leftarrow not F$

que define que, se não for possível provar $\neg F$ então F deverá ser verdadeiro.

Utilizando esta abordagem é possível criar um programa em lógica a partir de uma teoria abductiva (P, Ab) adicionado ao programa P para todos os literais L na lista de abduzíveis Ab duas regras com a seguinte forma:

1. $L \leftarrow not \neg L$
2. $\neg L \leftarrow not L$

3 Planeamento abductivo com o Event Calculus

De modo a representar e a raciocinar acerca de acções e eventos foi utilizado o Event Calculus com algumas alterações propostas por Eshghi ([4]) e Missiaen ([9]). As seguintes regras (propostas por Missiaen) são utilizadas para descrever as propriedades que são verdadeiras num dado instante de tempo.

$$holds\_at(P,T) \leftarrow happens(E), initiates(E,P), succeeds(E), E < T, persists(E,P,T). \quad (1)$$

$$persists(E,P,T) \leftarrow not\ clipped(E,P,T). \quad (2)$$

$$clipped(E,P,T) \leftarrow happens(C), terminates(C,P), succeeds(C), not\ out(C,E,T). \quad (3)$$

$$out(C,E,T) \leftarrow (T = C; T < C; C < E). \quad (4)$$

Estas regras definem que a propriedade P é verdadeira no instante de tempo T se houve um evento que aconteceu antes de T, se o evento iniciou P e se P persiste pelo menos até T. P, iniciado pelo evento E, persiste até T se não houver outro evento que termine P nesse intervalo.

De modo a inferir os planos dos utilizadores é necessário abduzir eventos (relacionados com as acções) e a sua ontologia temporal. O conjunto de predicados abduzíveis neste sistema é composto por:

Ab = {happens/1, act/2, </2}

O conjunto de abduzíveis permite a abdução dos eventos que podem explicar os efeitos observados e a sua relação temporal.

A representação das acções pode ser efectuada através de um mecanismo de tradução que, para cada regra da forma

- A causes F if P_1, ..., P_n

indicando que a acção A causa o efeito F se as pré-condições P_1, ..., P_n forem verdadeiras, produz as seguintes regras:

1. succeeds(E) $\leftarrow$ act(E, A), holds-at(P_1, E), ..., holds_at(P_n, E).
2. initiates(E, F) $\leftarrow$ act(E, A).

Este esquema de tradução indica que um determinado evento E associado com uma dada acção A poderá acontecer (ou ter acontecido) se as pré-condições forem verdadeiras no instante T e, como consequência, a propriedade F passará a ser verdadeira.

4 Exemplos

4.1 Diálogos sobre comboios

Neste exemplo (adaptado de [7]) é apresentado um diálogo sobre comboios onde a informação fornecida pelo passageiro começa por estar incompleta, tendo o sistema que encontrar uma justificação para o pedido feito, fornacendo a informaccão que lhe parece mais adequada para o plano que assumiu como sendo o do passageiro. Em seguida, o passageiro levanta um problema extra para o qual o sistema deve encontrar a melhor soluccão.

- Passageiro: O das 8:50 para Montreal?
- Empregado: 8:50 para Montreal? Linha 7.
- Passageiro: Onde é a linha?
- Empregado: Descendo à sua esquerda. Segunda à esquerda.
- Passageiro: Ok.

De modo a suportar este exemplo, o sistema tem de ter uma definição prévia da biblioteca de acções (acções do domínio, actos de fala) necessárias às inferências. Foram utilizadas as acções definidas em [7] que foram transformadas em regras de programação em lógica através do mecanismo de tradução referido anteriormente.

Nomeadamente, foram definidas as seguintes acções dependentes do domínio:

- Primeira regra:

goto(agent, location, time) *causes* at(agent, location, time).

Esta regra significa que a acção de um agente ir para um determinado local num dado tempo causa o efeito de ele se encontrar nesse local nesse dado instante. De acordo com a metodologia descrita daria origem a:

$$succeeds(E) \leftarrow act(E, goto(agent, location, time)), \quad (5)$$

$$initiates(E, at(agent, location, time)) \quad (6)$$

- Segunda regra:

meet(agent, arriveTrain) *if* goto(agent, gate(arriveTrain), time(arriveTrain)).

Esta regra significa que se um dado agente for para a linha de chegada de um comboio à hora da sua chegada, então ele encontrá-lo-á e a regra dará origem a:

$$succeeds(E) \leftarrow act(E, meet(agent, arriveTrain)), \quad (7)$$
$$holds\_at(goto(agent, gate(arriveTrain), time(arriveTrain)), E)$$

- Terceira regra:

board(agent, departTrain) *if*

goto(agent, gate(departTrain), time(departTrain)), geton(agent, departTrain).

Esta regra significa que se um dado agente for para uma linha de partida de um comboio à hora da sua partida e se entrar no comboio, então ele embarcará no comboio.

A regra dará origem a:

$$succeeds(E) \leftarrow act(E, board(agent, departTrain)), \quad (8)$$
$$holds\_at(goto(agent, gate(arriveTrain), time(arriveTrain)), E)$$
$$holds\_at(geton(agent, departTrain), E)$$

- Quarta regra:

take_train_trip(agent, departTrain, destination) *if*

buy_ticket(agent, clerk, ticket), board(agent, departTrain).

Esta regra significa que um agente vai numa viagem de comboio para um determinado destino se comprar um bilhete para esse destino e se embarcar nesse comboio.

A regra dará origem a:

$$succeeds(E) \leftarrow act(E, take\_train\_trip(agent, departTrain, destination), \quad (9)$$
$$holds\_at(buy\_ticket(agent, clerk, ticket), E)$$
$$holds\_at(board(agent, departTrain), E)$$

Além destas regras dependentes do domínio da aplicação, foram também definidas as acções relacionadas com os actos de fala:

- Primeira regra:

inform(speaker, hearer, proposition) *causes*

know(hearer, proposition),

know(hearer, know(speaker, proposition))

if

know(speaker, proposition),

surface_inform(speaker, hearer, proposition).

Esta regra significa que se um dado agente tiver conhecimento sobre uma dada proposição e se emitir um acto de fala acerca desse conhecimento (surface-inform), então essa acção de informar tem como efeito que o receptor ficará informado acerca da proposição e, além disso, saberá que o emissor conhece a referida propriedade. Esta regra dará origem a:

$$succeeds(E) \leftarrow act(E, inform(S, H, P), \quad (10)$$
$$holds\_at(know(S, P), E)$$
$$holds\_at(surface\_inform(S, H, P), E).$$
$$initiates(E, know(H, P)). \quad (11)$$
$$initiates(E, know(H, know(S, P)). \quad (12)$$

- Segunda regra:

informref(speaker, hearer, term, proposition) *causes*

knowref(hearer, term, proposition),

if

knowref(speaker, term, proposition),

know(hearer, proposition).

Esta regra significa que se um agente emissor tiver conhecimento sobre um termo de uma proposição e se o receptor conhecer a proposição, então a acção de informar uma referência acerca da proposição tem como efeito que o receptor ficará a conhecer esse termo da propriedade.

A regra dará origem a:

$$succeeds(E) \leftarrow act(E, informref(S, H, T, P)), \quad (13)$$
$$holds\_at(knowref(S, T, P), E),$$
$$holds\_at(know(hearer, proposition), E).$$

$$initiates(E, knowref(H, T, P)). \quad (14)$$

- Terceira regra:

informif(speaker, hearer, proposition) *causes*

knowif(hearer, proposition),

if

knowif(speaker, proposition).

Esta regra significa que se um agente emissor tiver conhecimento sobre a validade de uma proposição, então a acção de informar esse valor tem como efeito que o receptor ficará a conhecê-lo. Esta regra dará origem a:

$$succeeds(E) \leftarrow act(E, informif(S, H, P)), \quad (15)$$
$$holds\_at(knowif(S, P), E)$$

$$initiates(E, knowif(H, P)). \quad (16)$$

- Quarta regra:

request(speaker, hearer, action) *causes*

want(hearer, action),

know(hearer, want(speaker, action)),

if

want(speaker, action),

(surface-request(speaker, hearer, action) or

surface-request(speaker, hearer, informif(hearer, speaker, cando(hearer, action)) or

surface-inform(speaker, hearer, ¬ cando(speaker, action)) or

surface-inform(speaker, hearer, want(speaker, action))).

Esta regra significa que se um agente emissor pretender efectuar uma acção e se ele efectuar um dos seguintes actos de fala:

- Requerer ao receptor para efectuar a acção;
- Requerer ao receptor para o informar se ele receptor poderá efectuar a acção;
- Informar o receptor que não pode efectuar a acção;
- Informar o receptor que deseja que a acção se realize.

então o acto de fala tem como efeito que o receptor deseja efectuar a acção (diálogo cooperativo) e tem conhecimento de que o emissor deseja essa mesma acção.

Foram também definidas meta-acções, que possibilitam a inferência de meta-planos (planos sobre planos) de modo a poder-se raciocinar sobre o conteúdo do diálogo. Assim:

introduce-plan(speaker, hearer, action, plan) *causes*

want(hearer, plan),

next(action, plan),

if

request(speaker, hearer, action),

step(action, plan),

agent(action, hearer).

Esta regra significa que se houver o pedido de realização de uma acção que seja um passo de um dado plano, então a introdução desse plano tem como efeito que o receptor passa a desejar a referida acção e que essa acção será a próxima acção do plano.

Do mesmo modo, foram definidas as meta-acções *continue-plan*, *identify-parameter*, *correct-plan* e *modify-plan* (ver [7, 6]).

Após o processo de tradução de todas as acções, a primeira frase do passageiro criará os seguintes factos:

1. happens(e1)
2. act(e1, surface_request(pass, emp, infref(emp, pass, T, train(montreal))))
3. holds-at(knowref(pass, time(8:50), train(montreal)), e1)

descrevendo que o passageiro efectuou um pedido ao empregado para ser informado acerca de uma referência do comboio de Montreal.

Após este processo é possível efectuar inferências acerca de situações do presente, do passado e do futuro. Por exemplo, podem ser inferidas que acção pode ser consequência do referido acto de fala:

1. happens(e2), e1 < e2, act(e2, A),
2. A = act(e2, request(pass, emp, informref(emp, pass, T, train(montreal))))

Para inferir esta acção foi necessário utilizar a regra que define o acto de fala *request* e abduzir os seguintes factos:

1. happens(e0), e0 < e1,
2. act(e0, want(pass, informref(emp, pass, T, train(montreal))))

Isto é, houve uma acção anterior a $e1$ que iniciou a intenção do passageiro em ser informado acerca de uma dada propriedade do comboio para MontReal. É possível, agora, inferir quais as propriedades que são verdadeiras como consequência do acto de fala:

1. holds-at(X, e2)
2. X = want(emp, infref(emp, pass, T, tr(montreal)))
3. X = know(emp, want(pass, infref(emp, pass, T, tr(montreal))))

Esta inferência significa que o empregado quer informar o passageiro e que, além disso, ele tem conhecimento da informação que o passageiro requereu.

Por outro lado, é possível abduzir que acção, ou acções, puderam acontecer em consequência deste evento, bem como as propriedades que se poderão tornar verdadeiras:

1. happens(e3), e2 < e3,
2. holds-at(X, e3), act(e3, A)
3. A = infref(emp, pass, gate(7), tr(montreal))
4. X = knowref(pass, gate(7), tr(montreal))

É abduzido que o empregado informará o passageiro acerca da referida propriedade (é um diálogo cooperativo) e que o passageiro adquirirá essa informação (utilizando as regras 13 e 14).

Podemos ainda ir mais longe e tentar inferir um possível encadeamento de acções:

1. happens(e3), e2 < e3,
2. happens(e4), e3 < e4,
3. holds-at(X, e4), act(e4, A)
4. A = goto(pass, gate(7))
5. X = at(pass, gate(7))

Nesta situação é possível inferir que o passageiro irá para a linha do comboio para Montreal.

Continuando com o processo, é possível abduzir que o passageiro irá para a linha de comboio para embarcar no comboio (a desambiguação entre ir para a linha para embarcar ou para receber o comboio pode ser feita através do recurso a uma base de conhecimentos sobre os comboios, os seus horários e destinos). O passo seguinte no processo de inferência indicar-nos-ia que esta acção é um passo do meta-plano *introduce-plan*.

4.2 Mensagens de correio electrónico

Nestes exemplos (adaptados de [11]) são salientados os problemas originados por diálogos em que existem situações de erro. No primeiro existe um desfasamento entre o objectivo do utilizador e o seu plano, enquanto no segundo exemplo é o próprio objectivo que é impossível de atingir:

Exemplo 1:

- P: Queria falar com a Kathy. Qual é o número do telefone do hospital?
- R: Ela já foi para casa! O número de casa é o 555-8321.

Exemplo 2:

- P: Como é que posso proteger as minhas mensagens de correio electrónico? Não quero que o *system manager* as possa ler.
- R: Mesmo com protecções o *system manager* pode ler as mensagens!

Nesta abordagem o reconhecimento dos planos é feito através da inferência das crenças e das intenções dos agentes. Como operadores epistémicos que descrevem aspectos de estado mental dos agentes, foram definidos os seguintes ([1, 3]):

INT(a, α): o agente a tem como intenção α

BEL(a, p): o agente a acredita que p é verdadeiro

ACH(a, p): o agente a acredita que p será verdadeiro como consequência das acções que ele irá efectuar

EXP(a, p) $\leftarrow$ BEL(a, p) or ACH(a, p)

Acções mais complexas podem ser construídas a partir das acções atómicas através dos operadores TO e BY:

TO(α, p): o plano de efectuar α de modo a tornar p verdadeiro

BY(α, β, p): o plano de efectuar β fazendo α, sendo p verdadeiro

Como base do processo de inferência é necessário introduzir algumas regras que relacionam os diversos operadores epistémicos:

$\text{INT}(a, \text{TO}(\alpha, p)) \leftarrow \text{BEL}(a, \text{TO}(\alpha, p)), \text{INT}(a, \alpha), \text{ACH}(a, p)$

Esta regra indica que se um agente acredita que ao efectuar α torna p verdadeiro, se tem como intenção efectuar α e se quer tornar p verdadeiro, então tem como intenção efectuar α para tornar p verdadeiro.

Existe também a regra correspondente para a relação BY:

$\text{INT}(a, \text{BY}(\alpha, \beta, p)) \leftarrow \text{BEL}(a, \text{BY}(\alpha, \beta, p)), \text{INT}(a, \alpha), \text{INT}(a, \beta), \text{EXP}(a, p)$

Esta regra indica que se um agente acredita que ao efectuar α efectua β sendo p verdadeiro, se tem como intenção efectuar α e efectuar β e se espera que p seja verdadeiro, então tem como intenção efectuar α para efectuar β sendo p verdadeiro.

Além destas regras é necessário introduzir uma regra de integridade que indica a inconsistência entre BEL e ACH:

$\bot \leftarrow \text{BEL}(a, p), \text{ACH}(a, p)$

Ou seja é impossível acreditar em p e desejar que p venha a ser verdadeiro simultaneamente.

Com estas regras básicas e com o formalismo do Event Calculus definido anteriormente é possível abordar e raciocinar sobre os exemplos apresentados. Assim, no primeiro exemplo a intervenção do utilizador poderá dar origem aos seguintes factos:

1. happens(e1)
2. act(e1, request(user, system, infref(system, user, number(N), hospital)))
3. initiates(e1, int(user, talk(kathy)))

Com estes factos e, através do processo abdutivo é possível inferir que:

holds-at(e1, int(user, BY(phone(hospital), talk(kathy), at-hospital(kathy))))

Ou seja, a intenção do utilizador ao telefonar para o hospital é de falar com a Kathy, de acordo com a assunção de que ela está no hospital. De facto, para fazer esta inferência é necessário abduzir as seguintes acções:

1. exp(user, at-hospital(kathy))
2. bel(user, by(phone(hospital), talk(kathy), at-hospital(kathy)))

associadas a um evento anterior a $e1$.

Caso o sistema tenha conhecimento de que a Kathy está em casa (e não no hospital), será anulada a inferência efectuada pelo utilizador, e poderá ser efectuada a seguinte inferência:

1. exp(system, at-home(kathy))

2. holds-at(e1, bel(system, by(home(hospital), talk(kathy), at-home(kathy))))

utilizando as mesmas regras utilizadas para a inferência das intenções do utilizador. Tendo em conta estas inferências é possível gerar, por parte do sistema, uma resposta idêntica à efectuada no exemplo.

No segundo exemplo, a frase do utilizador dará origem aos seguintes factos:

- happens(e1)
- act(e1, request(user, system, infref(system, user, protect(N), mail)))
- initiates(e1, int(user, ¬ read(sm, mail)))

Neste caso o objectivo do utilizador é impossível de atingir de acordo com os factos e regras existentes. Nomeadamente, não existe nenhuma relação entre o acto de fala que indica a acção requerida e o objectivo indicado. Nestas situações o sistema indicará que é impossível atingir o objectivo requerido (dando, eventualmente, uma justificação adequada).

5 Conclusões e Trabalho Futuro

Neste artigo foi apresentado um sistema que suporta inferência abdutiva de acções e eventos recorrendo ao uso da programação em lógica extendida e ao Event Calculus. Este ambiente permite a implementação de mecanismos de remoção de contradições que possibilitam a inferência e o raciocínio sobre situações ambíguas em diálogos através da escolha da melhor interpretação para cada frase (a interpretação associada ao modelo mínimo).

Através deste formalismo, é possível abordar um conjunto bastante vasto de problemas relacionados com o reconhecimento de planos em diálogos. Nomeadamente é possível reconhecer e raciocinar sobre diálogos em que existem objectivos não directamente definidos (exemplos de D. Litman) ou em que os planos definidos para atingir os objectivos estão incorrectos (exemplo de M. Pollack).

Os resultados obtidos demonstram que com esta abordagem é possível efectuar a inferência de planos e atitudes num sistema de processamento de diálogos como o descrito por Lopes e Quaresma ([8, 12]) que tenta englobar numa arquitectura multicéfala as diferentes componentes necessárias a uma interacção robusta em língua natural. É necessário ainda avaliar comparativamente os resultados obtidos por Cohen e Levesque ([2]).

Referências

[1] Douglas E. Appelt and Martha E. Pollack. Weighted abduction for plan ascription. *User Modeling and User-Adapted Interaction*, 2(1), 1992.

[2] P. Cohen and H. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(3), 1990.

[3] P. Cohen, J. Morgan, and M. Pollack. *Intentions in Communication.* MIT Press, Cambridge, MA, 1990.

[4] Kave Eshghi. Abductive planning with event calculus. In *Proceedings of the International Conference on Logic Programming*, 1988.

[5] R. E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intellligence*, (2):189–208, 1971.

[6] D. Litman and J. Allen. A plan recognition model for subdialogues in conversations. *Cognitive Science*, (11):163–200, 1987.

[7] Diane J. Litman. *Plan Recognition and Discourse Analysis: An Integrated Approach for Understanding Dialogues.* PhD thesis, Dep. of Computer Science, University of Rochester, 1985.

[8] J. G. Lopes. Architecture for intentional participation of natural language interfaces in conversations. In C. Brown and G. Koch, editors, *Natural Language Understanding and Logic Programming III.* North Holland, 1991.

[9] Lode Missiaen. *Localized Abductive Planning with the Event Calculus.* PhD thesis, Univ. Leuven, 1991.

[10] Luís Moniz Pereira, José Júlio Alferes, and Joaquim Nunes Aparício. Contradiction removal semantics with explicit negation. In M. Masuch and L. Pólos, editors, *Knowledge Representation and Reasoning Under Uncertainty, Volume 808 of LNAI*, pages 91–106. Springer-Verlag, 1994.

[11] Martha E. Pollack. *Inferring Domain Plans in Question-Answering.* PhD thesis, Dep. of Computer and Information Science, University of Pennsylvania, 1986.

[12] P. Quaresma and J. G. Lopes. A two-headed architecture for intelligent multimedia man-machine interaction. In *B. de Boulay and V. Sgurev (eds). Artificial Intelligence V - methodology, systems, applications.* North Holland, 1992.

[13] Earl D. Sacerdoti. *A Structure for Plans and Behavior.* American Elsevier, New York, 1977.