Sistemas de Apoio à Aprendizagem

e

A Teoria Cognitiva ACT

Andréia Cristina G. Machion

Resumo

Este relatório apresenta um resumo dos estudos realizados durante o primeiro semestre de 2003 dentro da Disciplina Tópicos em Ciência da Computação. Os assuntos tratados são Inteligência Artificial aplicada à Educação (IA-ED) e Sistemas Tutores Cognitivos. A área de IA-ED preocupa-se com o desenvolvimento de sistemas computacionais para apoio ao processo ensino-aprendizagem, utilizando ferramentas da Inteligência Artificial. Os Sistemas Tutores Cognitivos são baseados na Teoria Cognitiva ACT (Anderson, 1983), a qual tenta descrever a cognição humana e, principalmente, como o ser humano aprende.

1 Introdução

A área de Sistemas Tutores Inteligentes (STI), também chamada de Inteligência Artificial aplicada à Educação, se preocupa com o desenvolvimento de sistemas baseados em computador para auxiliar a aprendizagem. Possuem um certo grau de autonomia durante sua interação com o usuário e, para isso precisam ter acesso a várias formas de representação de conhecimento e a vários tipos de raciocínio (Self, 1990). As técnicas utilizadas para representar conhecimento e os processos de raciocínio são técnicas estudadas pela área de Inteligência Artificial, daí o nome Inteligência Artificial aplicada à Educação (IA-ED).

Tais sistemas tiveram origem nos sistemas CAI (*Computer Aided Instruction*) e nos sistemas CBT (*Computer Based Training*), aos quais foi acrescentada alguma forma de raciocínio e de comportamento inteligente. Os primeiros sistemas tutores tinham o objetivo de simular um tutor humano, no entanto, nos dias de hoje o enfoque é outro, os sistemas atuais têm como objetivo servir como ferramentas que possam auxiliar no processo ensino-aprendizagem. Porém, o nome STI permanece.

Essa é uma área de pesquisa que apresenta algumas dificuldades, pois o conhecimento de como ensinar não é totalmente formalizado. Existem teorias para a representação do conhecimento e também teorias que tentam descrever como o ser humano aprende. A Teoria ACT é uma delas e tenta descrever a cognição humana através de sistemas de produção (Anderson, 1983). Um aspecto relevante dentro dessa teoria é como ocorre a aprendizagem. Os sistemas tutores cognitivos são sistemas tutores baseados na teoria cognitiva ACT.

O texto está dividido da seguinte maneira: a seção 2 descreve a arquitetura básica dos STI's e alguns conceitos importantes relativos ao processo ensino-aprendizagem; a seção três faz uma breve apresentação da Teoria Cognitiva ACT e suas hipóteses; a seção 4 traz a implementação da teoria dentro de uma arquitetura, ACT-R e a seção 5 expõe como os sistemas tutores cognitivos implementam os fundamentos da teoria bem como um exemplo; a seção 6 mostra alguns resultados e estudos empíricos desenvolvidos.

2 Sistemas Tutores Inteligentes e o Processo Ensino Aprendizagem

Os componentes de um STI clássico são (Wenger, 1987):

- módulo do domínio do conhecimento: conteúdo que está sendo ensinado e a representação desse conhecimento por um especialista no assunto;
- modelo do estudante: monitora o desempenho do estudante;
- módulo pedagógico: fornece um modelo para o processo de ensino;
- módulo de comunicação ou interface: interage com o estudante.

A interação entre esses componentes é mostrada na Figura 1. O **modelo do estudante** fornece dados para que o **módulo pedagógico** escolha a próxima ação instrucional que pode ser: exercícios de reforço, correção de um erro ou avanço no conteúdo. Para isso, o **módulo pedagógico** busca informações no **módulo do domínio de conhecimento** para obter o conteúdo a ser comunicado ao estudante bem como a forma de representá-lo. Beck et al. (1996) identifica essa representação como um quinto componente do modelo de Wenger - o **modelo do especialista**. Qualquer que seja a ação do **módulo pedagógico**, esta acontece através do **módulo de comunicação** ou interface.

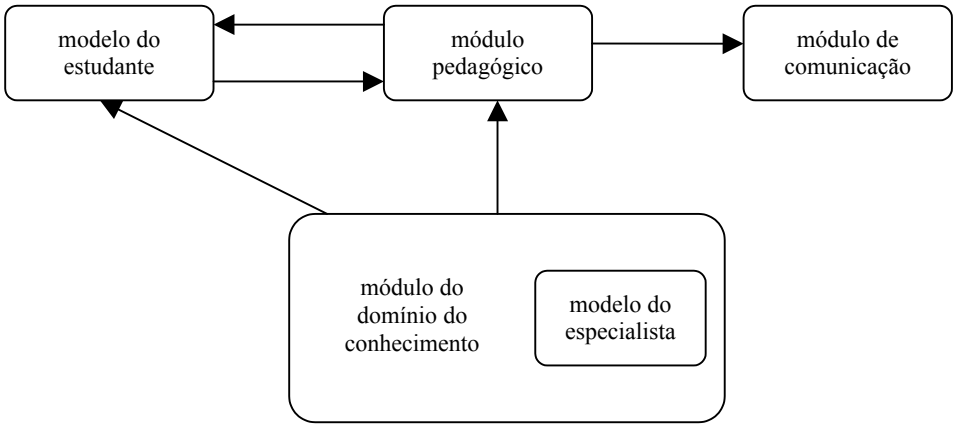

Figura 1: Interação entre os componentes de um STI

Antes de falarmos sobre cada um dos módulos e sobre as técnicas de IA utilizadas, é preciso considerar algumas características importantes sobre a área de educação. Existem dois níveis de interpretação :

- num sentido mais amplo, temos a educação informal, a cultura que nos é transmitida e que recebemos através dos nossos sentidos e que dirigem nosso comportamento;
- no sentido restrito, temos a educação formal que obtemos, em geral, em ambientes criados com o propósito específico de permitir e facilitar a aprendizagem.

O nível considerado neste trabalho será o nível restrito, a educação formal, pretende-se estudar métodos de apoio ao processo ensino-aprendizagem e, sendo assim, é preciso levantar algumas questões relativas a esse processo, entre elas, a **natureza do conhecimento**, a **natureza da aprendizagem**, o **nível de instrução**, **novas tecnologias** e **eficácia de um sistema STI**.

NATUREZA DO CONHECIMENTO

Para que seja possível construir um STI, é necessário decidir qual é a representação adequada para o conhecimento que se quer ensinar. É importante observar que nem sempre o conhecimento específico é representado da mesma forma que seria feito num Sistema Especialista de Inteligência Artificial. Existem várias filosofias tradicionais da educação que sugerem uma forma própria de representação do conhecimento, entre elas:

- **Objetivismo** - focaliza o conhecimento que deve ser aprendido, considerando que este já está construído e que o mundo é representado como entidades, suas propriedades e relacionamentos de forma precisa. As regras são passadas e cabe ao estudante armazená-las.
- **Construtivismo** - o aprendiz molda as informações que recebe e constrói o seu próprio conhecimento. A ênfase está em como o indivíduo estrutura o mundo, de forma que qualquer evento ou entidade pode ser visto de diferentes perspectivas por alunos diferentes.
- **Situacionismo** - compartilha algumas idéias com o construtivismo, defendendo o fato de que o indivíduo constrói o seu conhecimento, mas isso acontece quando ele interage com o meio. Situacionistas defendem que o conhecimento não é exatamente as representações criadas, mas a capacidade de criá-las.
- **Coneccionismo** - defende que o conhecimento esta representado na memória, através dos diferentes pesos atribuídos às ligações entre os milhares de neurônios.

Existe também o conhecimento de como ensinar, o qual não está totalmente formalizado e pode ser colocado na forma de regras ou simplesmente ser representado implicitamente pelo conjunto de decisões do projeto da interface com o aluno.

NATUREZA DA APRENDIZAGEM

Cada filosofia de representação do conhecimento implica, de uma certa forma, na maneira como ocorre aprendizagem. Qualquer sistema que auxilie no processo ensino-aprendizagem deve, então, incluir atividades que suportem várias maneiras de aprender tais como tentativa e erro, estudo de casos, experimentação, revisão de conceitos, aprendizado com atividade social etc..
Uma tentativa de ligar a teoria da aprendizagem aos sistemas de IAED é a que relaciona as teorias cognitivas aos sistemas tutores inteligentes, como por exemplo, a teoria ACT* e o sistema GREATERP (tutor de LISP para iniciantes) que será visto mais adiante.

NÍVEL DE INSTRUÇÃO

Qual é o nível de instrução que o sistema deve fornecer: instruir, tutorar, guiar ou treinar?. Esse aspecto refere-se à interação entre o sistema e o estudante e tem como objetivo definir **como** o sistema vai interagir com o estudante, o que nem sempre está bem definido pois, como visto anteriormente, esse é o conhecimento mais complexo de ser formalizado. Além disso, assim como o aluno tem diversas formas de aprender, cada professor tem um estilo próprio para ensinar.

A decisão a ser tomada é quando e como o sistema deve intervir. Esta é uma questão aberta, não só porque a área de IAED é nova, mas também porque novas teorias sobre o processo ensino-aprendizagem surgem continuamente, determinando novos papéis para o tutor e para o estudante dentro do processo. Existe forte tendência a descentralizar a figura do professor, tornando o processo mais colaborativo, ou mesmo mais individualizado no que diz respeito ao ritmo de aprendizagem de cada um.

NOVAS TECNOLOGIAS

Novas tecnologias surgem todos os dias e cada uma delas traz uma série de inovações que proporcionam, não só mostrar algum assunto de forma motivadora ou mais fácil de visualizar, mas também meios de interação e colaboração permitindo uma troca de informações muito mais rapidamente. Porém, é preciso não perder o foco do trabalho que é criar um ambiente inteligente que proporcione aprendizagem, ou seja, não basta incorporar novas tecnologias, é preciso considerar como elas realmente vão ajudar.

EFICÁCIA DE UM STI

Essa é uma das etapas mais difíceis do processo de desenvolvimento de Sistemas Tutores. Existem várias propostas de avaliação que muitas vezes são consideradas insatisfatórias. Existem duas estratégias de avaliação de um STI:

- avaliação do sistema como um todo e dos resultados que ele obtém do ponto de vista educacional e,
- avaliação dos componentes do sistema para que eles possam ser melhorados.

3 A Teoria Cognitiva ACT

ACT (*Atomic Component of Thought*) é uma teoria geral sobre a cognição humana com ênfase na aquisição de conhecimento (Anderson, 1983). Ela começou a ser desenvolvida em 1976 (Anderson, 1976) e uma de suas primeiras versões a ACT* foi proposta em 1983. A versão atual que ficou

pronta em 1993 é conhecida como ACT-R (Anderson, 1993). A esta nova versão foram acrescentados novos módulos, que estudam o desenvolvimento motor, principalmente visual e manual. Tanto na teoria ACT* quanto na sua sucessora, a teoria ACT-R, existe uma distinção entre o conhecimento declarativo e o conhecimento procedimental, originada na Inteligência Artificial (IA) (Winograd, 1975) e que foi modificada para ser usada pela psicologia.

As características principais da teoria ACT* que não se alteraram são:

- *Distinção entre procedimental e declarativo*. O conhecimento declarativo é *o que* sabemos enquanto que o procedimental diz respeito a *como* usamos o conhecimento adquirido.
- *Construção do conhecimento*. A teoria assume que regras de como usar o conhecimento só podem ser aprendidas aplicando-se o conhecimento declarativo ao contexto da resolução de um determinado problema.
- *Reforço*. Só é possível tornar o conhecimento sólido (tanto declarativo quanto procedimental) com a prática.

Um conceito importante utilizado nessa teoria são os **Sistemas de Produção da Inteligência Artificial**, que são compostos por um **conjunto de dados**, um conjunto de **regras de produção** e um sistema de controle (Nilsson, 1982). Essas regras têm sido uma das maneiras mais utilizadas para representar as bases de conhecimento em sistemas especialistas. Geralmente elas são da forma

`SE condição verdadeira ENTÃO ação`*`(ões)`*

Existem duas estratégias de raciocínio para as regras de produção que podem ser executadas por máquinas de inferências, *forward chaining* e *backward chaining*, em ambas o caminho de inferência é o que é procurado entre o estado atual e o objetivo.
Quando *forward chaining* é utilizado, a condição inicial conhecida é entrada para o lado esquerdo da regra para que seja possível gerar um conjunto de conclusões. Esse conjunto é checado para verificar se ele casa com o objetivo procurado e, em caso negativo, essas conclusões são utilizadas como entradas em regras subsequentes que devem gerar novas conclusões.
O *backward chaining* é utilizado quando as regras selecionadas têm o seu lado direito casando com o objetivo. O lado esquerdo é considerado então como objetivo intermediário que precisa ser satisfeito e regras que satisfazem essa condição são ativadas. Este processo tenta encontrar uma situação em que todos os lados esquerdos sejam verdadeiros no sistema.

Na Teoria ACT, as regras de produção representam o conhecimento procedimental enquanto que o conjunto de dados representa o conhecimento declarativo.

A Figura 2 ilustra a arquitetura de um sistema de produção básico da Teoria ACT[1] desenvolvida por Anderson (1983). Segundo ele, existem três memórias: a **de trabalho**, a **de produção** e a **declarativa**. Sendo que a primeira modela a **memória humana de curta duração** e as últimas, parte da **memória de longa duração** (Russel & Norvig, 2002). A memória de trabalho contém informações de acesso imediato, que podem ser informações recuperadas da memória de longa duração declarativa, estruturas temporárias recebidas e codificadas do ambiente e também ações das produções, ou seja, ela contém conhecimento declarativo, permanente ou temporário, que se encontra em estado ativo.

[1] Para simplificar a leitura, a partir deste ponto a teoria será chamada de Teoria ACT, pois os tópicos aqui estudados são comuns a todas as suas versões.

A memória declarativa consiste de fatos que não estão necessariamente ligados a um determinado contexto, podemos ter, por exemplo, uma abstração do tipo

```
Dois ângulos cujas medidas somam 180° são chamados suplementares.
```

Por outro lado, uma produção na memória procedimental é codificada de forma domínio-específica. Um exemplo de produção derivada da definição acima pode ser:

```
SE    queremos provar que a medida de um ângulo 1 é igual à medida
de um ângulo 2
      e o ângulo 1 é suplementar ao ângulo 3
ENTÃO tente provar que o ângulo 2 é suplementar ao ângulo 3.
```

Note que a codificação de conhecimento através de regras permite a utilização de **variáveis**. No exemplo, temos as variáveis ângulo 1, ângulo 2 e ângulo 3.

Os principais processos dentro dessa arquitetura são:

- **percepção**: deposita informações vindas do ambiente na memória de trabalho;
- **ação**: transforma os comandos na memória de trabalho em comportamento;
- **armazenamento**: pode criar registros permanentes na memória declarativa do conteúdo da memória de trabalho e também pode aumentar a força dos registros já existentes na memória declarativa;
- **recuperação**: recupera informações da memória declarativa para a memória de trabalho;
- **casamento**: os dados da memória de trabalho são colocadas em correspondência com as condições das produções;
- **execução**: deposita as ações das produções comparadas e satisfeitas da memória de produções na memória de trabalho.

Os processos de casamento e execução juntos formam a **aplicação** de produções. Note que este processo reflete-se na memória de produção, isto ocorre porque novas produções são aprendidas a partir do histórico das produções existentes, o que na teoria ACT significa que só é possível aprender fazendo.

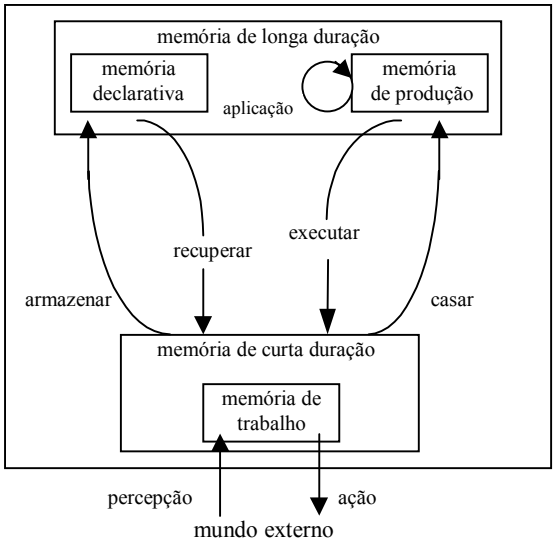

Figura 2: Representação do sistema de produção ACT

A teoria combina conhecimento declarativo na forma de redes semânticas com conhecimento procedimental na forma de regras de produção. A aprendizagem se dá através da formação de novas regras. Existem quatro mecanismos de aprendizagem dentro da teoria ACT (Anderson, 1983) que resumimos a seguir:

1. **Interpretação de fatos**. Informações vindas do ambiente são depositadas na memória de trabalho e estas podem ser armazenadas de forma permanente na memória de longa duração declarativa. Essas informações pode ser recuperadas e interpretadas posteriormente. Este mecanismo tem como vantagem a flexibilidade, porém seu custo é alto em termos de armazenamento na memória de trabalho e tempo de busca.

2. **Reforço**. Toda vez que um conhecimento, armazenado na memória declarativa ou na procedimental, é usado, ele está sendo reforçado e isso aumenta a sua chance de ser utilizado novamente. Uma regra de produção é utilizada toda vez que ela é selecionada na resolução de um conflito e disparada. Um fato é utilizado quando ele é comparado com a condição de uma regra que foi disparada. O reforço não altera o estado do conhecimento, ele só aumenta a chance do conhecimento ser utilizado novamente, ou seja, o sistema se concentra no conhecimento que é freqüentemente utilizado.

3. **Compilação do Conhecimento**. Pode ser dividida em dois subprocessos:
   - **Composição**: toma uma seqüência de produções utilizadas na resolução de um problema e aglutina todas numa única regra cujo efeito será o mesmo da seqüência.

   - **Construção de Regras**: assume a separação entre o objetivo e o contexto de uma regra, eliminando a referência ao conhecimento declarativo do domínio usado para a resolução de um problema e monta as conseqüências do conhecimento numa regra de produção para um domínio específico, ou seja, constrói uma versão generalizada das produções para que não seja necessária a recuperação de conhecimento declarativo específico de domínio para a memória de trabalho.

   Como exemplo, considere os procedimentos abaixo para a discagem de um número, Anderson, (1976), percebeu, através de experimentos, que quando discamos um número de telefone freqüentemente, desenvolvemos um procedimento especial para fazê-lo, de forma que nem precisamos mais acessar as informações declarativas, isto é, o número do telefone.

```
P1        SE    o objetivo for discar Vnúmero
                e Vdígito1 for o primeiro dígito de Vnúmero
          ENTÃO discar Vdígito1.

P2        SE    o objetivo for discar Vnúmero
                e Vdígito1 já tiver sido discado
                e Vdígito2 for o número seguinte a Vdígito1 em Vnúmero
          ENTÃO discar Vdígito2.
```

   A composição dessas regras resulta em:

```
P1&P2     SE    o objetivo for discar Vnúmero
                e Vdígito1 for o primeiro dígito de Vnúmero
                e Vdígito2 for o número seguinte a Vdígito1
          ENTÃO discar Vdígito1 e então Vdígito2.
```

   Uma composição como essa reduz o número de aplicações de produções para a realização de uma tarefa, porém ela ainda necessita que informações sejam recuperadas da memória

declarativa para a memória de trabalho, no caso os dígitos do número do telefone. A criação de produções elimina as cláusulas na condição que necessitam dessa recuperação. Para esse exemplo, as segunda e terceira cláusulas serão eliminadas e substituídas por valores, por exemplo, se a tarefa for discar o número da Maria que é 432-2815 várias vezes, as variáveis locais em `P1&P2` serão substituídas e teremos a seguinte produção:

```
P1&P2*    SE    o objetivo for discar número_da_Maria
          ENTÃO discar 4 e 3.
```

Continuando o processo de compilação de conhecimento, obtemos a seguinte produção:

```
P*        SE    o objetivo for discar número_da_Maria
          ENTÃO discar 432285.
```

Uma observação importante é que a construção de novas produções não implica na perda de produções antigas nem da representação do conhecimento declarativo. A perda de conhecimento tanto declarativo quanto procedimental só ocorre quando esses caem em esquecimento.

As produções originais `P1` e `P2` podem ser utilizadas para quaisquer números enquanto que `P*` só para o número da Maria. Portanto, todas as regras se mantém, e a que for mais útil é a que permanece.

4. **Generalização e Discriminação.** São relacionados à aprendizagem indutiva, isto é, aprendizagem que infere novo conhecimento a partir do conhecimento recebido não só pela substituição de variáveis mas também encontrando restrições que são verdadeiras tanto para a produção a ser generalizada/discriminada quanto para a generalização/discriminação. Uma discriminação tenta restringir o escopo de aplicação de uma produção para circunstâncias adequadas. Esses mecanismos tentam extrair de exemplos de sucesso e de falha o que caracteriza a aplicação de uma regra. Porém, por se tratarem de processos indutivos, eles podem levar a erros, pode ocorrer super generalização ou discriminações inúteis.

A Teoria ACT afirma que o conhecimento declarativo pode ser assimilado pelo ser humano de forma bastante rápida, sem o compromisso de saber como é possível utilizá-lo, ele pode ser armazenado na memória através de uma instrução dada ou por leitura ou por palavras. Por outro lado, o conhecimento procedimental só pode ser adquirido através da prática do conhecimento declarativo, ele é caracterizado pelo fato de representar fielmente o conhecimento de maneira eficiente e específica quanto ao conteúdo. O conhecimento procedimental é um subproduto da interpretação do conhecimento declarativo.

4 A Arquitetura ACT-R

O desenvolvimento em psicologia cognitiva é paralelo ao trabalho em IA. Uma das maiores contribuições da IA é demonstrar que teorias de estruturas internas do conhecimento e processos de aprendizagem podem ser rigorosamente demonstrados (Anderson, 1989). A cognição humana é bastante complexa, portanto arquiteturas cognitivas devem ser capazes de fazer previsões também complexas. Geralmente a cognição não é linear, o que torna métodos matemáticos analíticos infactíveis para realizar tais previsões e, se métodos analíticos falham, a simulação é o próximo método a ser utilizado. A modelagem cognitiva não tem a ambição de construir máquinas inteligentes conscientes, ela tenta somente entender melhor a inteligência humana usando a

simulação por computador [Taatgen, 1999]. Geralmente, uma arquitetura é um algoritmo que simula uma teoria cognitiva não linear. Este algoritmo pode ser usado para fazer previsões para tarefas específicas dentro de domínios específicos. Uma teoria cognitiva baseada em simulação precisa de restrições e limites devido às limitações da computação. Para construir um modelo cognitivo (Figura 3), é necessário seguir alguns passos:

1º - fazer uma **análise da tarefa**
- determinar qual o **conhecimento necessário** para executar tal tarefa;
- saber quais são os passos para executar tal tarefa.

2º - escolher uma **arquitetura cognitiva** ou uma teoria cognitiva ou uma ferramenta de modelagem
- especificar o conhecimento acumulado na análise da tarefa na representação da forma da arquitetura.

3º - comparar
- ajustar os **parâmetros** para o casamento das **previsões** do modelo com os **dados**.

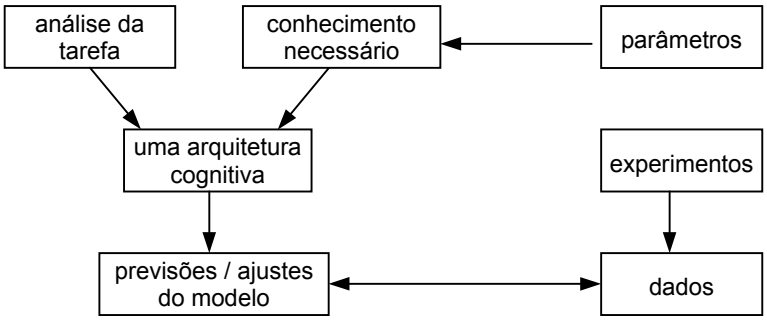

Figura 3: Modelo para construção de um Modelo Cognitivo

Os objetivos do Projeto ACT-R são:
- explicar o máximo possível sobre cognição (aprendizagem), utilizando uma única teoria;
- construir uma arquitetura cognitiva, um ambiente de simulação, que simule o comportamento humano e para isso, as previsões da simulação devem ter a melhor granularidade possível quanto ao tempo de reação, erros, escolhas, curvas de aprendizado, movimento dos olhos etc..

A Arquitetura ACT-R (Controle Adaptativo do Pensamento Racional) (Anderson, 1993; Anderson & Lebiere, 1998) se baseia principalmente na distinção entre as memórias declarativa e procedimental (Anderson, 1976) e tem foco na aprendizagem. Segundo a análise racional, cada componente da arquitetura cognitiva é otimizado de acordo com a demanda do ambiente, observadas as limitações computacionais, ou seja, se quisermos saber como um aspecto particular da arquitetura deve funcionar, devemos olhar primeiro como esse aspecto pode funcionar de maneira ótima no ambiente simulado. Anderson (1990) relaciona essa otimalidade à evolução do sistema. Um exemplo deste princípio é a maneira como a escolha é implementada no ACT-R: sempre que existe uma escolha entre qual estratégia usar ou qual elemento da memória recuperar, o sistema vai escolher aquele que tem o maior ganho esperado, que é a escolha que tem o menor custo esperado e a maior probabilidade de sucesso, como será visto adiante.

A arquitetura ACT-R (ACT-R, 2002) não tem memória de trabalho separada, ela usa a memória declarativa em conjunto com um conceito de **ativação** para armazenamento dos fatos *short-term* (de curta duração na memória). Para manter um histórico do contexto, ACT-R usa uma pilha de objetivos e o elemento do topo da pilha é chamado o **foco da atenção** - um ponteiro para um

elemento na memória declarativa que representa o objetivo atual. Novos objetivos podem ser inseridos na pilha e o atual pode ser removido.

ACT-R é uma arquitetura híbrida, pois possui dois níveis de descrição. O nível **simbólico** é o sistema de produção e o **sub-simbólico** é representado por um conjunto de processos paralelos que pode ser resumido em um conjunto de equações que controlam os processos do nível simbólico.

NÍVEL SIMBÓLICO

As representações na memória são discretas e o processamento é feito através do ciclo típico *reconhece-executa_ação* dos sistemas de produção, com a memória declarativa fazendo o papel de memória de trabalho.

As estruturas nessa memória são chamadas ***chunks***. Um *chunk* armazena informações na forma proposicional e pode conter um fato, um objetivo (atual ou antigo) ou uma informação sobre percepção, além disso, ele sempre tem um ***id*** e seus atributos podem referenciar outros *chunks*. Um exemplo de um *chunk* que armazena o objetivo "executar 6 + 2" e a resposta ainda não foi encontrada é:

```
GOAL26
ISA       ADDITION
ADDEND1   SIX
ADDEND2   TWO
ANSWER    NIL
```

Neste exemplo, `ADDEND1`, `ADDEND2` e `ANSWER` são os *slots* do *chunk* chamado `GOAL26` que tem id (isa) `ADDITION` enquanto que `SIX` e `TWO` são os atributos (valores) para os dois primeiros *slots*. `SIX` e `TWO` são referências a outros *chunks* na memória declarativa. Como o *slot* `ANSWER` tem valor NIL, significa que a resposta ainda não foi encontrada. Assumindo que este *chunk* seja o objetivo atual, se o ACT-R deve encontrar o valor para o *slot* `ANSWER` e a atenção está focada em algum outro objetivo, `GOAL26` vai se tornar parte da memória declarativa e retornar que 6 + 2 = 8. Mais tarde, esse fato vai ser ativado para uso subsequente.

Informações procedimentais são representadas na memória procedimental pelas regras de produção. Como vimos, uma regra de produção tem dois componentes básicos: a parte **condicional (LHS)** e a parte da **ação (RHS)**. A primeira deve ter padrões que casam com o objetivo corrente e possivelmente com outros elementos na memória declarativa, enquanto que a segunda pode modificar os valores de *slots* nos objetivos e/ou criar sub-objetivos. Como exemplo, uma regra que tenta resolver um problema de subtração chamando um *chunk* para adição pode ser:

```
IF    o objetivo é subtrair num2 de num1 e não há resposta
AND   existe um chunk para adição num2 mais num3 igual a num1
THEN  coloque num3 no slot answer do objetivo.
```

Este exemplo também mostra um aspecto importante das regras de produção - as variáveis, neste caso, *`num1`*, *`num2`* e *`num3`* são variáveis, isto significa que esta regra pode achar a resposta para qualquer problema de subtração se o *chunk* de adição necessário estiver disponível.

NÍVEL SUB-SIMBÓLICO

O nível simbólico, como visto anteriormente, fornece os blocos de construção básicos da arquitetura ACT-R. Usando somente este nível já é possível construir modelos interessantes de tarefas para as quais existe um conjunto de regras a serem aplicadas bem definido. Porém, a utilização só desse

nível deixa um grande número de detalhes sem especificações. A parte principal que é delegada ao nível sub-simbólico é a escolha. Uma escolha é feita sempre que existe mais que uma regra de produção aplicável a um objetivo ou quando existe mais que um *chunk* cujo padrão casa com a primeira parte de uma regra de produção. Outros aspectos que o nível sub-simbólico trata são erros, esquecimentos, previsões e tempos de latência.

A escolha é baseada na:
- probabilidade de que um determinado chunk é necessário;
- utilidade (ganho esperado) de uma certa regra de produção;

para isso, cada regra de produção e cada *chunk* tem um certo número de parâmetros.

No caso dos *chunks*, na memória declarativa, estes parâmetros são utilizados para calcular uma estimativa da chance deles serem ativados no contexto corrente. Esta estimativa, chamada de **ativação** tem dois componentes: uma **ativação de nível básico** que representa a relevância do *chunk* por ele mesmo, e **ativação de contexto** que o relaciona aos $chunks_j$ do objetivo atual:

$$Ai = Bi + \sum_j WjSji$$

onde

A_i = ativação do $chunk_i$
B_i = ativação de nível base do $chunk_i$
Sji = medida de associação entre os elementos i e j
Wj = peso dado ao elemento j.

Por exemplo, no caso do problema de subtração 8-2=?, o fato de que 8 e 2 são parte do contexto aumenta a probabilidade de *chunks* associados a 8 e a 2 serem ativados. Neste caso 2+6=8 receberá ativação extra através de ambos, 2 e 8.

A **ativação** de um *chunk* tem várias conseqüências para seu processamento. Se existe mais que um *chunk* cujo padrão casa com a regra de produção, o *chunk* com a **ativação** mais alta é escolhido. Diferenças entre os níveis de ativação (ruídos) também podem levar a disparidades, por exemplo, um *chunk* com ativação mais alta cujos padrões não casam completamente com a regra de produção pode ser é selecionado. A **ativação** também desempenha um papel com relação à latência: quanto menor a **ativação** de um *chunk*, mais tempo leva para ativá-lo. Para evitar que um tempo de ativação passe da ordem de um segundo, define-se um limite máximo. Resumindo, a **ativação** de um *chunk* determina:
- o tempo de ativação,
- se ele pode mesmo ser ativado,
- qual é o escolhido quando mais que um casa com o padrão requerido,
- se existir um que não casa com o padrão requerido, mas tiver ativação alta (ruído), a sua escolha.

A escolha entre regras de produção na memória procedimental é determinada pela estimativa de ganho esperado chamada **utilidade** e os parâmetros empregados são:

P_i = probabilidade estimada de sucesso da regra i
C_i = custo estimado de sucesso (em segundos) da regra i
G = valor do objetivo.

Note que os parâmetros P_i e C_i são de cada regra, enquanto que G é global. O valor da **utilidade** é, então, dado por:

$$Ui = Pi * G - Ci$$

Para ser possível estimar esses valores, ACT-R mantém os parâmetros para cada regra de produção. A **utilidade** também determina a latência de disparo de uma regra de produção: regras com maior **utilidade** levam menos tempo para serem disparadas. Quando existe mais que uma regra que atinge o objetivo, a que tiver maior **utilidade** é escolhida.

APRENDIZAGEM

Uma vez que a arquitetura ACT-R tem dois sistemas de memória distintos com dois níveis de descrição, diferentes mecanismos de aprendizagem são implementados para representar o conhecimento e os seus parâmetros. No nível simbólico, os mecanismos de aprendizagem especificam como novos *chunks* e novas regras são adicionados às memórias declarativa e procedimental respectivamente. No nível sub-simbólico, eles mudam os valores dos parâmetros de **ativação** de um *chunk* e de **utilidade** de uma regra. Nenhum objeto é removido da memória, eles podem se tornar virtualmente irrecuperáveis.

No nível simbólico, um novo *chunk* na memória declarativa tem duas origens possíveis: ou ele é um objeto de percepção, ou ele é um *chunk* criado internamente pelo processamento de um objetivo. Estes últimos são sempre velhos objetivos que passam pelo processo de **reforço** da teoria.
Aprender uma nova regra de produção é um processo mais complexo (**compilação de conhecimento**), as regras são aprendidas por exemplos que são estruturados em forma de um *chunk* de dependência, um tipo especial de *chunk* que aponta para todos os componentes necessários à montagem da nova regra de produção. Uma vez que um *chunk* de dependência não é um *chunk* de percepção, ele deve ter sido um objetivo antigo. Então, para aprender uma nova regra, um objetivo de dependência deve ser inserido na pilha de objetivos e, depois de processado, ele deve ser removido e o mecanismo de compilação de produção faz a sua generalização para uma regra de produção. Resumindo, novas regras de produção são aprendidas através da combinação de várias regras existentes em uma, eliminando-se a recuperação na memória declarativa do processo (**construção de regras e composição**).

Como no nível sub-simbólico os parâmetros estimam as propriedades de certos elementos da base de conhecimento, a aprendizagem neste nível tem como objetivo ajustar as estimativas experimentalmente. O princípio que guia estes ajustes é o **Teorema de Bayes**, que diz que uma nova estimativa para um parâmetro é baseada no seu valor anterior e na experiência atual.

Como visto, a *ativação de nível básico* de um *chunk* estima a probabilidade dele ser ativado sem levar em consideração o contexto atual, portanto cada vez que um *chunk* é ativado ou recriado, ela é aumentada e com o passar do tempo sem ativação, ela vai diminuindo. Existem duas estratégias para essa aprendizagem:

- recuperar um fato da memória declarativa:
  - é rápido, mas necessita desse fato na memória
  - a velocidade é baseada na freqüência de uso anterior
- verificar respostas através de contagem
  - é lento mas sempre funciona
  - leva mais tempo ainda quando os operandos são grandes

Na memória procedimental acontece um ajuste no ganho esperado e no custo de cada regra, baseado nos sucessos e nas falhas dos experimentos anteriores:

$$P = \frac{sucessos}{sucessos + falhas}$$

e

$$C = \frac{\sum custos}{sucessos + falhas}$$

sendo que os parâmetros para experimentos iniciais são fixos.

5 Sistemas Tutores Cognitivos

O interesse inicial do grupo de pesquisa ACT em produzir sistemas tutores inteligentes era testar a teoria cognitiva, ou seja, estudar como as pessoas aprendem ou adquirem competências. Desta forma, o sistema é baseado em regras de produção que modelam como um estudante resolve determinados problemas. Os primeiros sistemas tutores construídos incorporam algumas idéias básicas (Anderson et al., 1995), a saber:

- **Modelo**: existe um modelo de regras de produção da habilidade básica a ser aprendida incorporada ao STI. Este modelo deve executar a tarefa da maneira que se espera que um aluno execute - é o modelo do aluno ideal.
- **Ações no caminho correto**: ações corretas dos estudantes são reconhecidas se estas fazem parte do conjunto de soluções que o modelo possui e, quando isso acontece, o STI não interfere no trabalho do aluno.
- **Ações no caminho incorreto**: nos sistemas tutores mais recentes, se o estudante segue por um caminho fora do conjunto de caminhos possíveis, ele é orientado para que ele consiga voltar para um caminho correto. Os primeiros eram mais rígidos e exigiam que o aluno se mantivesse num caminho correto o tempo todo.
- **Respostas sobre erros e sistema de ajuda**: os sistemas tutores possuem dois tipos de instruções de interação. Um acontece quando o estudante comete um erro e, neste caso, uma mensagem que explica porque a ação é um erro é gerada. O outro, quando o estudante pede ajuda ao sistema e aí uma mensagem que pode guiá-lo para a solução correta é apresentada. Cada um dos tipos de mensagem é sensível ao contexto.

Dentro do modelo clássico de um STI, o primeiro item representa o módulo do conhecimento do domínio, o segundo e o terceiro, o módulo pedagógico e o último, a interface.

No início do desenvolvimento desses tutores percebeu-se também que deveria haver uma conexão mais estreita entre os sistemas e a Teoria ACT. Anderson et al. (1987) extraíram, então, oito princípios da teoria para projetos de tutores:

Princípio 1: Representar as competências do estudante como um conjunto de regras de produção. A idéia principal é que o material a ser estudado seja apresentado como um modelo correto e fiel da competência que deve ser aprendida. O modelo cognitivo permite ajustar o conteúdo e interpretar as ações do estudante. No entanto, esse princípio não determina a interface, nem a forma de interação, nem como ocorre o progresso do estudante dentro do material, tudo isso depende de uma teoria de como as regras de produção são assimiladas.

Princípio 2: Comunicar a estrutura do objetivo para guiar a resolução do problema. Uma das hipóteses da Teoria ACT é que a resolução de problemas envolve a sua decomposição em sub-objetivos e uma observação relevante é que, em muitos assuntos onde o aluno apresenta dificuldades, tanto o problema quanto o roteiro de resolução não foram apresentados adequadamente. Portanto, a comunicação dos objetivos devem fazer parte do material instrucional, por isso as interfaces foram desenvolvidas de forma que as estruturas do objetivo fiquem bem claras. Um exemplo de sucesso desse princípio é a estrutura de grafos utilizada na demonstração de teoremas da geometria.

Princípio 3: Fornecer instruções no contexto de resolução do problema. É baseado na especificidade do aprendizado (Anderson, 1990) e uma dificuldade que surge é determinar a hora certa de apresentar a instrução, se antes ou durante a resolução do problema.

Princípio 4: Propiciar um entendimento abstrato sobre o conteúdo do problema estudado. Este princípio foi motivado pelo fato de que um estudante tende a tornar o conhecimento adquirido em algo muito específico, ou seja, ele não é capaz de generalizar uma regra de produção. Como os problemas são baseados em situações reais, tentou-se reforçar a abstração correta através das mensagens de erro e de ajuda.

Princípio 5: Minimizar a quantidade de informações que precisam estar na memória de trabalho. Toda instrução necessária para o aprendizado de novas regras devem estar ativas na memória e, portanto, se houver informações desnecessárias, essas podem interferir no processo de aprendizagem. Este princípio tem por objetivo apresentar somente as instruções necessárias para a resolução de um determinado problema, ou seja, apresentar uma instrução específica para um componente do problema somente quando as anteriores tiverem sido assimiladas. Tal procedimento direciona para um projeto de currículo em que pequenas porções de informações são apresentadas de cada vez, o que leva o aluno a aprender a lidar com situações complexas gradualmente.

Princípio 6: Fornecer resposta imediata quando acontecer erros. Na Teoria ACT*, a construção de novas regras de produção é baseada em toda a resolução do problema, portanto quanto maior a demora em mostrar um erro, maior a propagação do mesmo, já na sua sucessora, a ACT-R, a construção de novas regras de produção é baseada somente na resposta final do problema, portanto não importa os erros no meio do caminho, logo não seria necessário mostrar o erro imediatamente após a sua ocorrência, pois para o sistema isso não faz diferença. Mas, experimentalmente, observou-se que, para o aluno, isso diminui o tempo de aprendizagem.

Princípio 7: Ajustar a granularidade da instrução ao nível de aprendizagem. Este princípio baseado num operador da teoria ACT* supõe que regras de produção simples podem compor regras maiores, isto é, uma regra que executa num único passo o que foi executado em vários passos anteriormente. Isto é interessante para analisar a resolução do aluno em unidades maiores.

Princípio 8: Facilitar aproximações sucessivas ao objetivo. No início do aprendizado de alguma nova competência, o aluno não consegue executar todos os passos para atingir o objetivo, então o tutor completa essas lacunas. Conforme o aluno vai praticando, o número de lacunas vai diminuindo e o sistema interfere cada vez menos até que ele não precise mais executar qualquer ação. Experimentalmente, este princípio tem funcionado bem.

Alguns desses princípios (3, 6, 7 e 8) são similares às idéias *behavioristas* que guiaram projetos de tutores dessa linha (Bunderson & Faust, 1976; Gagné & Briggs, 1974). Outros (1, 2, 4 e 5) são usados para se obter diferentes representações de cada competência que se quer ensinar. Os aspectos

nos quais esses princípios adicionam alguma coisa à abordagem *behaviorista* refletem justamente as diferenças nas representações do conhecimento.

O fato dos tutores cognitivos incorporarem os modelos cognitivos das competências não implica, necessariamente, que eles sempre se comportem de maneira diferente daqueles baseados nas idéias *behavioristas*, depende também do domínio. Num tutor para ensinar a escrever palavras, por exemplo, onde o objetivo é a memorização, o modelo cognitivo seria bem próximo do modelo *behaviorista,* no entanto, os tutores cognitivos foram construídos para domínios mais complexos onde levam a diferentes estratégias instrucionais.

A abordagem utilizada nos sistema tutores cognitivos é chamada "*model tracing*", pois tenta relacionar a resolução do estudante a alguma seqüência de disparo das regras do modelo cognitivo do sistema. Este é um problema computacional difícil, pois o número de possibilidades é muito grande. Uma forma encontrada para contornar esse problema foi, em cada ação, monitorar se o aluno está num caminho possível e, quando alguma ação gera ambigüidades, um menu que o ajuda a solucionar o problema é apresentado. Este método de guiar o estudante mais uma interface que acompanha cada passo dele permite reduzir o espaço de possibilidades e, com isso, é possível acompanhar seus passos em tempo real.

Para que seja possível implementar essa metodologia, é necessário criar todas as regras que estarão envolvidas na resolução de um problema e também as regras para os erros. Uma forma de implementar essas últimas é observar os erros que os estudantes cometem, tentar entender a sua origem e então codificá-las no sistema uma a uma. Em trabalhos mais recentes alguns dos erros já podem ser gerados baseados em princípios e regras, por exemplo no tutor de álgebra, observa-se que os alunos esquecem freqüentemente da etapa de distribuição (Matz, 1982).

Observando o conjunto de regras que descrevem o comportamento do estudante, o projeto dos tutores cognitivos pode ser divididos em três módulos, os quais estão de acordo com o modelo clássico (Wenger, 1987):

- **modelo do estudante**: o qual pode acompanhar o comportamento do estudante através de um conjunto de regras não determinístico;
- **módulo pedagógico**: que engloba as regras para interação com o estudante, para a seleção de problemas e para atualização do modelo do estudante;
- **interface**: tem a responsabilidade de interagir com o estudante.

5.1. MODELO DO ESTUDANTE

A sua principal tarefa é passar para o módulo pedagógico uma interpretação de um comportamento em termos das várias seqüências de regras de produção que poderiam ter causado tal comportamento. A metodologia utilizada é realizar uma busca *forward* no conjunto de regras e verificar quais caminhos produzem o comportamento procurado. Existem dois fatores importantes que influenciam na implementação do módulo do estudante:

- **Não determinismo**: é a maior dificuldade na implementação do *model tracing*. Ele aparece sempre que muitas produções no módulo do estudante geram a mesma saída. Um exemplo no tutor de álgebra é quando o aluno diz que deve-se aplicar a distributiva e existem vários termos onde isso pode ocorrer. Um caso especial é quando a produção gera uma resposta aberta, por exemplo quando um aluno está fazendo algum cálculo ou um planejamento "de cabeça", além do número de possibilidades para o caminho que está sendo percorrido crescer exponencialmente, existe o fato de que não se sabe exatamente o que o estudante estaria pensando e, portanto, não se pode calcular o *feedback* correto. Neste caso, a ação natural é

questionar o aluno para forçar uma associação da sua resposta com alguma etapa do caminho de solução. Por outro lado, esta seria uma interface pesada para o aluno, já que ela estaria interrompendo o seu trabalho constantemente e, às vezes, desnecessariamente para ele. Um outro problema do não determinismo é que pequenos erros e entendimento errado de um conceito podem produzir o mesmo comportamento. Neste caso, o módulo do estudante deve ser capaz de fornecer as duas interpretações para o tutor, para que ele possa avaliar as probabilidades de um e outro e decidir qual a correção a ser aplicada.

- **Eficiência dos Sistemas de Produção**: executar um sistema de produção em tempo real pode criar sérios problemas. Um aluno não vai ficar esperando o sistema descobrir qual é o caminho que ele está seguindo nem mudar o ritmo do seu trabalho para ajudar o programa de diagnose. Apesar das vantagens de um sistema de produção, essa não é a maneira mais rápida de se resolver um problema. Os problemas computacionais inerentes aos sistemas de produção são aumentados em sistemas tutores pelas seguintes razões:
  - A granularidade da modelagem é menor do que a que seria necessária em sistemas especialistas e a complexidade necessária para os padrões das regras para que seja possível detectar erros é considerável.
  - O sistema tem que considerar todas as produções necessárias em qualquer ponto para poder reconhecer todos os próximos passos que o estudante poderá gerar.
  - Nem sempre é possível saber em qual dos caminhos corretos o estudante está, então o sistema deve ser usado não deterministicamente para que seja possível rastrear um conjunto de caminhos até que as ambigüidades possam ser eliminadas.

Os sistemas de produção implementados utilizam variações do algoritmo RETE (Forgy, 1982), para casamento de múltiplos padrões e, embora existam bons resultados, ainda há problemas que precisam ser tratados. Um exemplo simples pode ser visto no tutor de álgebra quando se está ensinando fatoração:

```
2AB + 4A
2BA + 4A
```

Um aluno enxergaria mais facilmente a possibilidade de fatoração no primeiro exemplo, por outro lado num curso mais avançado em que fatoração não é o objetivo, não seria necessário considerar as duas representações (Anderson *et al.*, 1995). Além disso, problemas de eficiência têm impacto nos tópicos abordados:

- Os problemas tendem a ter custo maior quanto maior for o seu tamanho, mesmo que a base de conhecimento necessária seja a mesma. Isto ocorre porque a memória de trabalho aumenta assim como o não determinismo.
- O progresso para tópicos mais avançados é limitado pela carga computacional adicionada pela modelagem adequada do entendimento desses tópicos.
- As interações durante a resolução do problema fica limitada pela necessidade de diminuir o não determinismo, por exemplo, o tutor pode forçar uma interpretação ao invés de esperar que o aluno gere a solução necessária para eliminar a ambigüidade.

Se verificarmos todas as seqüências possíveis de produção que podem ser geradas pelos modelos, podemos dizer que o espaço do problema é finito e, embora a cardinalidade seja grande, isso ocorre porque são diferentes permutações de passos independentes na resolução de um problema. Isso significa que esses passos não precisam ser gerados dinamicamente para serem interpretados, pode-se fazer isso antes da utilização do programa e, durante a resolução do problema, comparar com a solução do aluno.

Existem outras vantagens quando se tem o espaço do problema compilado antes da sessão do tutor, isto permite que ele saiba onde o estudante pode chegar com uma determinada resposta.
Geralmente, num tutor de provas, uma regra de produção pode ser favorecida pelo modelo ideal, mas não necessariamente direciona para a solução. Por exemplo, existem problemas de geometria que mesmo os especialistas fazem inferências que não são utilizadas na solução final do problema. Esta é uma heurística que obtém sucesso na maioria das vezes, mas nem sempre.

Conforme o aluno vai resolvendo os exercícios, o tutor utiliza um procedimento Bayesiano para estimar a probabilidade de que ele tenha aprendido cada regra do modelo cognitivo. Enquanto o aluno não atinge um determinado índice para cada regra (estabelecido experimentalmente em 0,975), o tutor não avança no conteúdo, ao contrário, ele fornece mais exercícios relacionados àquele assunto. Este processo de verificação da aprendizagem é chamado ***Knowledge Tracing*** (acompanhamento do conhecimento)
A verificação da aprendizagem assume **um modelo de aprendizagem com apenas dois estados**. Cada regra de produção ou está no **estado não aprendida** ou no **estado aprendida**. Uma regra pode passar do primeiro para o segundo estado ou pelo processo de aprendizagem ou através da aplicação da mesma na prática. Além disso, não há "esquecimento", ou seja, não é possível passar do segundo para o primeiro estado. O desempenho na aplicação de uma regra é ditado pelo seu estado, de maneira probabilística, isto é, se uma regra estiver no estado aprendida, ainda assim, o aluno pode cometer erros, da mesma forma se a regra estiver no outro estado ele pode fazer um chute correto. Conforme o estudante vai praticando, o tutor mantém uma estimativa de aprendizado para cada regra e em cada oportunidade onde esta regra é aplicada, essa estimativa é atualizada, de acordo com a resposta do aluno.

5.2. MÓDULO PEDAGÓGICO

Durante o desenvolvimento dos tutores foi possível separar a estratégia pedagógica do domínio de conhecimento, o que torna a metodologia muito mais genérica. O domínio de conhecimento é inserido dentro do modelo do estudante e da interface enquanto que o módulo pedagógico relaciona os dois e controla a interação. O desenvolvimento do módulo pedagógico (Anderson et al., 1990) pressupõe que é possível criar estratégias de ensino para os sistemas que sejam livres de domínio, pois a Teoria ACT para aquisição de conhecimento acredita que os princípios básicos de aprendizagem sejam livres de domínio. Portanto, esse módulo diz respeito a:

- quais produções que podem ser aplicadas numa determinada circunstância, sem se preocupar com a semântica interna da produção;
- quais respostas que o estudante gerou e se essas respostas estão de acordo com as respostas que o modelo ideal geraria, sem se preocupar com o significado dessas respostas;
- quais perfis de diálogo ligados a cada produção, sem se importar com o significado deles.

Controle de *Feedback*
Um dos princípios para o desenvolvimento de tutores cognitivos diz respeito ao tempo de *feedback*. Segundo esse princípio, os sistemas devem fornecer mensagens de erro imediatamente após um deles ter sido detectado e exigir sua correção também imediata, pois isso minimiza problemas de ambigüidades. Além disso, existem evidências psicológicas que um *feedback* é tão mais efetivo quanto mais próximo ele estiver do erro (Anderson, 1983), pois é mais fácil para o estudante analisar o que o levou àquele erro e também um *feedback* imediato torna a aprendizagem mais eficiente porque evita que o estudante construa toda uma solução em cima de um suposição errada (Lewis & Anderson, 1985). Mas não existem só vantagens nessa estratégia, alguns problemas podem surgir:

- o *feedback* deve ser cuidadosamente elaborado para forçar o estudante a pensar, ou seja, deve-se força-lo a construir a nova resposta e não dar a resposta para ele;
- às vezes o estudante poderia ter chegado ao erro sozinho se tivesse tido um pouco mais de tempo, o que teria resultado numa aprendizagem mais efetiva (Anderson, 1983);
- interrupções constantes podem atrapalhar ao invés de ajudar;
- pode ser difícil explicar uma escolha errada sem que o aluno tenha terminado o seu raciocínio.

O único modelo de estudante utilizado nos tutores é um modelo genérico composto por todos os movimentos corretos e incorretos que um aluno pode fazer. Ao longo do tempo, o sistema é capaz de processar todas as regras de produção que um aluno pode executar, sejam elas corretas ou não. No entanto, diante de um erro do estudante, o *feedback* é sempre o mesmo, independente da sua história. Isto ocorre porque a teoria (Anderson, 1989) supõe que as pessoas aprendem basicamente da mesma forma.

Além do tipo de controle que fornece *feedback* imediato, outros três foram implementados com o objetivo de estudar qual seria o que proporciona maior nível de aprendizado:
- Sem qualquer tipo de interrupção no desenvolvimento do estudante, este último é que pergunta, quando ele quiser saber, se a sua solução está correta. O tutor responde somente sim ou não e não há explicações para os erros.
- Assinalamento do erro (*flag*) assim que ele ocorre, em negrito na tela, mas sem qualquer tipo de mensagem espontânea. O estudante pode perguntar qual é o tipo de erro, obtendo, neste caso, a mesma resposta que seria dada no controle padrão automático ou ele pode tentar corrigir o erro sozinho. Ele pode também continuar no seu desenvolvimento e voltar mais tarde ou corrigir imediatamente.
- Sem interrupção no trabalho do estudante, porém ao longo do desenvolvimento é possível pedir ajuda (*on demand*) ou *feedback* sobre os erros.

Foram realizados testes utilizando o tutor de LISP e testes posteriores com lápis e papel para os grupos submetidos a cada um dos quatro tipos de controle e os resultados podem ser conferidos na seguinte tabela.

Tipo de *feedback*	**índice de acerto**	**tempo médio por exercício (s)**
imediato	55%	250
flag	55%	390
on demand	58%	480
nenhum	43%	750

Conteúdo do *Feedback*
Uma característica importante de um tutor é o que ele diz ao estudante durante a resolução de um problema. Existem duas ocasiões em que essa interação acontece, a primeira quando o estudante pede ajuda e a outra quando o tutor detecta algum erro na solução. Para avaliar os benefícios de uma explicação para os erros além do seu apontamento, várias medidas foram feitas:
- índice de erros por regra estudada: 15% quando existe *feedback* explicativo contra 22% quando não existe;
- correções certas dos erros: 65% quando existe *feedback* explicativo contra 38% quando não existe.

5.3. INTERFACE

O sucesso ou o fracasso de um STI depende de como a interface é projetada (Anderson, 1990). É importante ter um sistema que deixa bem claro ao estudante o objetivo da resolução do problema e

onde estão os seus erros. O sistema deve também minimizar a carga na memória de trabalho do aluno, expondo os conceitos necessários, para que ele não precise lembrar de tudo que ele já aprendeu.

5.4. EXEMPLO - O TUTOR DE GEOMETRIA

Esse tutor é baseado em estudos sobre estratégias de resolução de problemas que suportam a geração de provas em geometria. Ele utiliza grafos de demonstração e também a abordagem "*model tracing*" (Anderson,1990). A Figura 4 mostra como o problema é apresentado ao aluno: no topo da figura está o objetivo, o que deve ser provado, embaixo estão os dados do problema e no canto esquerdo superior um diagrama. O sistema pede que ele aponte quais as premissas e a regra que devem ser utilizadas e a seguir o estudante deve digitar a conclusão que ele tirou, após o que a tela é atualizada. A seqüência de premissas mais a regra de inferência mais a conclusão formam um passo de inferência.

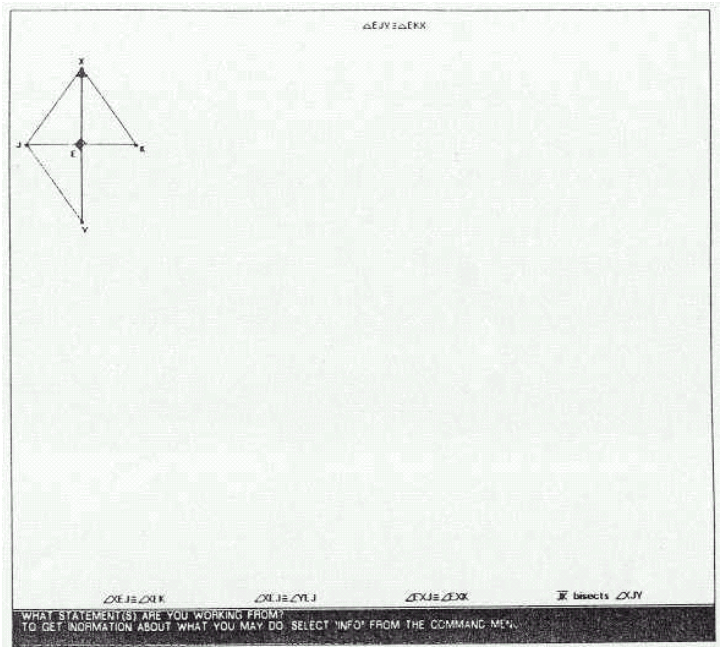

Figura 4: Uma tela inicial do Tutor de Geometria

A Figura 5 mostra a tela no momento em que o aluno selecionou a definição de bisseção para aplicar a premissa $\overline{JK}$ *faz a bisseção do ângulo* $\angle XJY$ mas ainda não digitou a conclusão. Então aparece um menu no lado esquerdo contendo as relações e símbolos da Geometria que permite a digitação da sua conclusão. Basta que o estudante aponte para os símbolos no menu e para os pontos no diagrama para formar a sua afirmativa, no caso $\angle XJK \cong \angle KJY$. A necessidade do aluno apontar para os pontos no diagrama serve para o sistema tutor verificar se o aluno sabe a que ele está se referindo.

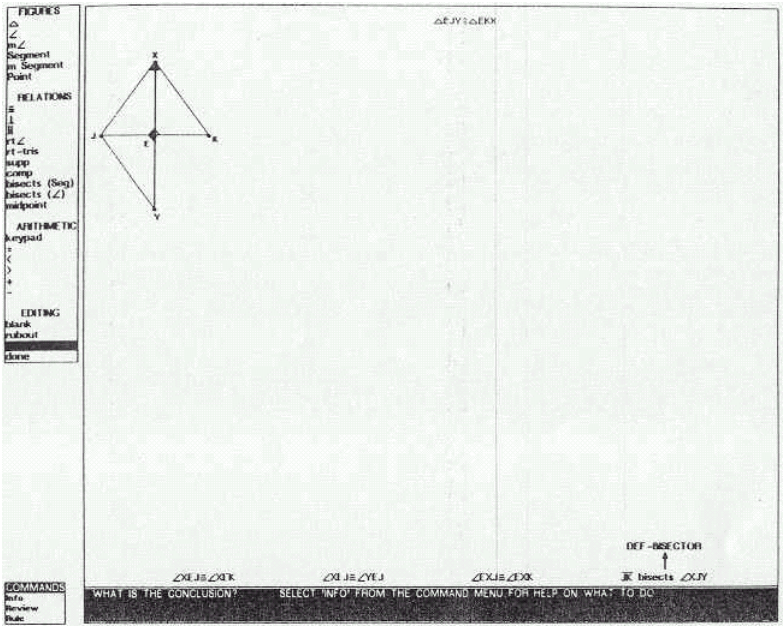

Figura 5: Tela de configuração depois que o aluno selecionou as premissas, a regra mas não digitou a conclusão ainda

A Figura 6 mostra um ponto mais adiante na resolução do problema onde o aluno completou a inferência da bisseção e adicionou uma inferência válida sobre transitividade, mas essa última não será utilizada na prova. O tutor interfere, avisando sobre a inferência inútil na situação do problema e o aluno começa a chutar possíveis soluções. O seu último chute, que está na figura, é a tentativa de aplicar a regra *ângulo-lado-ângulo* (ALA) às premissas $\angle EJX \cong \angle EJY$ e $\angle EXJ \cong \angle EXK$. O tutor mostra que a regra ALA necessita de três premissas e, portanto, esta regra não é apropriada. Como o estudante está se mostrando perdido, o sistema tutor dá uma dica do que deve ser feito: para provar que $\Delta EJY \cong \Delta EKX$, deve-se provar que $\Delta EJY \cong \Delta EJX$ e $\Delta EJX \cong \Delta EKX$ e depois aplicar transitividade. A maneira como o tutor faz isso é destacando a conclusão e pedindo para que o aluno utilize uma busca *backward* para procurar uma regra e um conjunto de premissas que levaram àquela conclusão. Se for necessário, o sistema também mostra ao aluno como a transitividade leva à conclusão do exercício. Cabe ao estudante, então, mostrar as duas congruências entre os respsctivos triângulos.

A Figura 7 mostra o estágio no qual o aluno conseguiu mostrar a primeira das congruências, faltando apenas uma. É possível, neste ponto da resolução, que o estudante utilize tanto busca *back* quanto *forward*, pois a estrutura de resolução está bem clara.

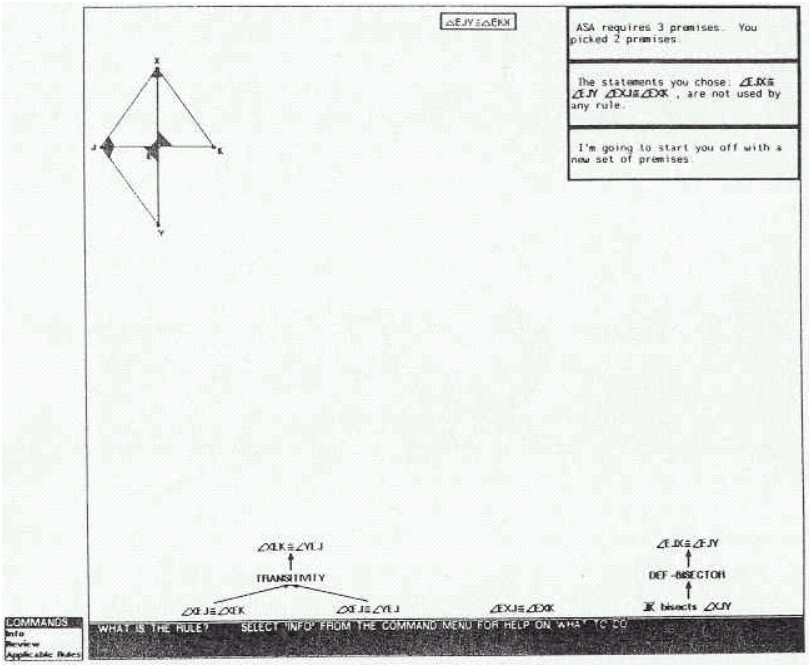

Figura 6: estudante tentou utilizar ALA às premissas $\angle EJX \cong \angle EJY, \angle EXH \cong \angle EXK$

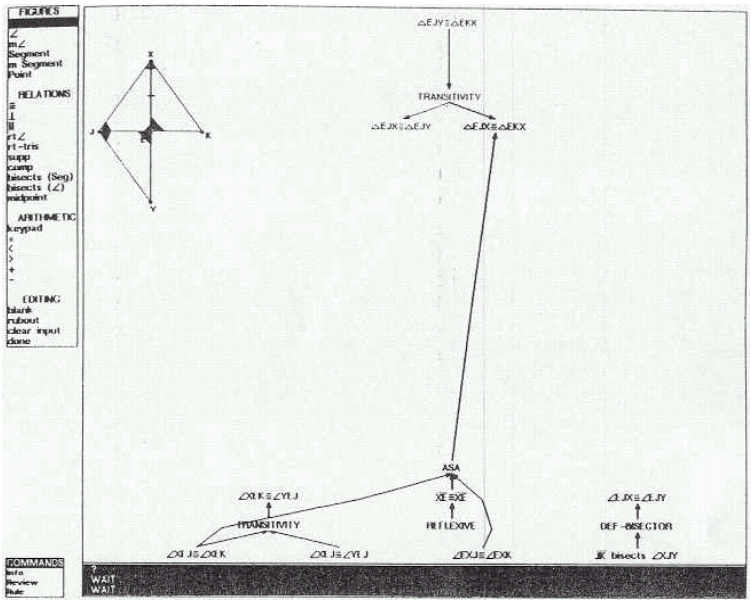

Figura 7: O estudante conseguiu provar uma das congruências entre os triângulos

A Figura 8 mostra a prova completa, na qual temos uma estrutura de grafos conectando os dados à conclusão da resolução. Os estudantes acham que esse tipo de estrutura facilita a aprendizagem (Anderson et al, 1990) por duas razões, primeiro porque mostra como as inferências se combinam para chegar a uma prova e segundo, porque mostra explicitamente e de maneira concreta a representação da busca inerente a um problema de prova. No exemplo, podemos ver até as inferências que foram feitas mas não foram utilizadas.

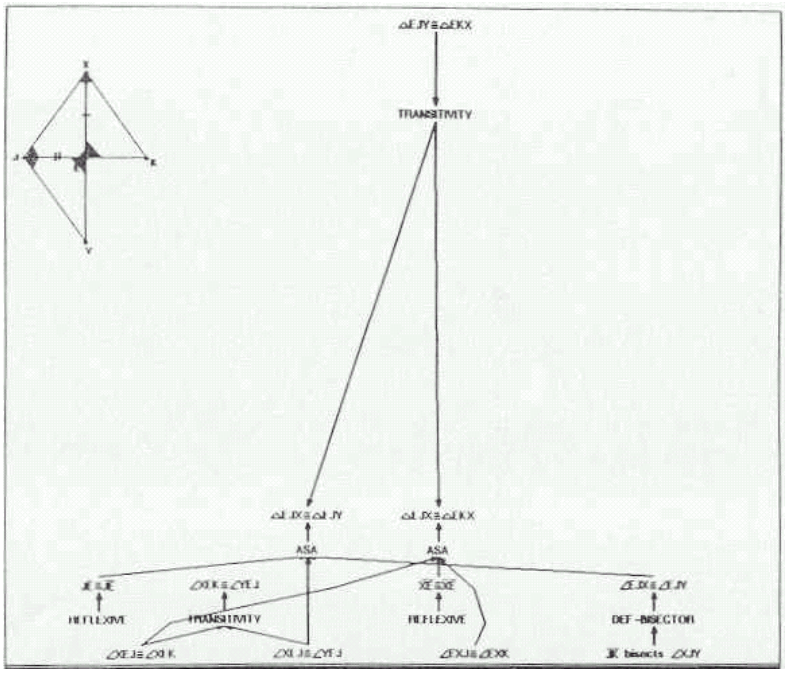

Figura 8: O problema resolvido

6 Estudos e Resultados Experimentais

APRENDIZAGEM POR COMPONENTES

Uma das controvérsias entre a Ciência Cognitiva e a Educação é se é possível tomar uma competência complexa, quebrá-la em componentes menores e entender a aprendizagem da mesma entendendo a aprendizagem de cada componente. As experiências realizadas mostraram que sim. Anderson (1989) concluiu que existem quatro fatores críticos:

- Prática das regras de produção. Quanto mais um estudante usa uma regra de produção na resolução de exercícios, melhor o seu desempenho em aplicar aquela regra. Na Teoria ACT, este fato se deve à idéia de REFORÇO.
- Exploração dos detalhes. Tanto no tutor de LISP quanto no tutor de geometria percebeu-se que o tempo e a precisão da aplicação de regras melhoram conforme o estudante vai explorando detalhes específicos do problema e resolvendo cada um deles. Na Teoria ACT, isto é atribuído ao reforço da representação declarativa do problema através de acessos repetidos.
- Fator de Aprendizagem. Analisando o desempenho dos estudantes é possível perceber que cada um aplica novas regras com graus de sucesso diferentes. Isto pode ser pelo simples fato de que cada pessoa é diferente e/ou também pela quantidade de horas estudos que cada um realizou.
- Fator de Retenção. Neste caso também, cada estudante tem desempenho diferente, só que agora ao aplicar regras vistas em lições anteriores. Novamente pode ser só por diferenças individuais e/ou horas de estudos.

METODOLOGIA DE DESENVOLVIMENTO

A equipe ACT, depois de várias implementações e testes desenvolveu uma metodologia de desenvolvimento de Tutores Cognitivos composta de cinco estágios bem definidos (Anderson, Corbet, Koedinger & Pelletier, 1995):

1. Construção da Interface e o Problema da Transferência

A interface entre o estudante e o computador será o mundo real no qual a resolução de problemas pelo estudante deve ocorrer. A habilidade que está sendo ensinada é a resolução de problemas numa determinada interface. Portanto, a interface deve ser projetada antes de identificar as regras de produção para a mesma. Ao se desenhar uma interface, deve-se ter em mente o domínio para o qual a habilidade será transferida, pois embora muitas escolas ainda esperem que os seus alunos tenham proficiência na manipulação algébrica com papel e lápis, existe uma forte tendência para a habilidade-objetivo envolver o uso de softwares de computadores. Portanto, parte da competência que tenta se ensinar no tutor de álgebra, por exemplo é como construir planilhas, como trabalhar com pacotes gráficos e de manipulação algébrica. Depois de identificar as habilidades-objetivo, deve-se projetar uma interface que possibilite a transferência para essas habilidades.

Existem duas maneiras de se construir a interface na Arquitetura ACT-R durante o desenvolvimento de um tutor cognitivo - a primeira é construir a interface utilizando as características e facilidades do pacote, a outra é utilizar um software específico e associá-lo à arquitetura. De qualquer forma é preciso observar que:

- as ações tomadas na interface devem ser repassadas ao tutor, pois ele precisa saber que ações o estudante tomou para poder acompanhá-lo e guiá-lo durante a resolução do problema;
- o tutor precisa saber das conseqüências de uma determinada ação para o estado da interface, pois o modelo cognitivo precisa manter na sua memória de trabalho a representação da interface que o estudante está visualizando;
- tutor deve ser capaz de executar ações na interface por si próprio.

Respeitando-se essas três restrições, existe bastante flexibilidade nos tipos de interface que se pode utilizar. Um tipo que tem alcançado sucesso é a interface que utiliza um software específico e ensina o estudante a resolver problemas nesse software, com isso mesmo estando fora do tutor, o estudante ainda tem um ambiente de resolução de problemas conhecido e útil, de tal forma que o problema de transferência é minimizado. Por exemplo, ensinar a resolver problemas algébricos utilizando o *Excel* ou problemas geométricos utilizando o *Sketchpad*.

2. Especificação do Currículo (Conteúdo)

A especificação do conteúdo consiste na identificação do tipo de problema que se espera que o estudante seja capaz de resolver num determinado domínio e as restrições na resolução. O tutor acompanha o desempenho do aluno em diversas produções (*knowledge tracing*) para guiá-lo através do conteúdo, promovendo-o de um nível para outro conforme o seu progresso nas regras. O professor que estiver utilizando o tutor poderá ativar ou desativar essas características de acordo com a situação, a qualquer momento.

3. Modelagem Cognitiva ou Modelagem em Regras de Produção

A construção de um ambiente para resolução de problemas e um conjunto de problemas bem determinados para serem resolvidos nesse ambiente é o mesmo que a especificação comportamental da competência-objetivo. A maior tarefa da modelagem cognitiva é então descobrir o que tal competência significa em termos de um conjunto de regras de produção que serão capazes de promover essa competência comportamental de uma maneira cognitiva adequada. Isto é exatamente a construção do modelo ideal de estudante. Esse modelo é capaz de enviar as ações que constituem a resolução correta para o problema em questão para a interface e, como na maioria dos casos existem várias soluções possíveis, o modelo do estudante ideal deve ser capaz de gerar todas as soluções de maneira não determinística.

As regras de produção interagem com as informações na memória de trabalho, que podem ser de dois tipos: informações sobre o estado corrente do problema e representações sobre o objetivo corrente.

Uma vez estabelecidas as regras de produção para a resolução do problema, é necessário compará-las às respostas do estudante, isto requer o acréscimo de testes que comparem as respostas do estudante com as regras, para que o tutor seja capaz de determinar qual regra foi disparada na cabeça do estudante. Se houver ambigüidades, menus de seleção serão ativados a partir de perfis também armazenados com cada regra.

4. Projeto das Instruções Declarativas

Parte das competências num determinado domínio podem vir de instruções declarativas dadas fora do tutor. Essas podem ser conceitos gerais ou informações que servirão como base de onde as regras de produção foram extraídas. Elas podem vir de textos extras ou de aulas expositivas sobre o assunto, sempre existe um material escrito que acompanha o tutor. Uma ferramenta que tem sido utilizada ultimamente com sucesso são os hipertextos, que são disponibilizados pela regra de produção que deve ser aprendida naquela seção. Essas instruções mostram exemplos que ilustram as regras salientando sempre os aspectos importantes e os conceitos. É importante lembrar que o tutor é minimalista e expõe somente o necessário para a aprendizagem daquela regra. Internamente, existem dois tipos de instruções declarativas nos tutores:

- mensagens de erro: são exibidas quando o estudante comete um e apresentam alguma informação útil, isto requer regras de produção que vão por caminhos errados às quais estão anexadas essas mensagens;
- mensagens de ajuda: são exibidas quando o estudante pede ajuda ou quando o tutor detecta a sua necessidade, neste caso, a mensagem está anexada à regra que deveria ter sido disparada naquele estado.

Um problema que surge é o uso exagerado da ajuda por parte de alguns alunos, que consequentemente aprendem pouco. Uma forma encontrada para tentar minimizar esse problema é associar a disponibilização das mensagens de ajuda ao *knowledge tracing*. Além disso, dois aspectos ainda estão sendo estudados, o primeiro refere-se à exibição das mensagens de ajuda espontaneamente, sem que o aluno tenha solicitado e o outro à forma como as mensagens devem ser apresentadas: uma dica com o conteúdo todo ou várias mensagens com conteúdo gradual.

5. Aplicação Prática em Salas de Aula

Dependendo do assunto e do tipo de classe com os quais se está trabalhando, a utilização do tutor pode ser diferente. Em turmas de programação na CMU, o tutor é utilizado de forma individual, de maneira que cada aluno determina o seu ritmo. Isso acontece devido a características bem peculiares: o material é restrito ao curso, as turmas são adultas e têm familiaridade com computadores, o tipo de assunto exige que cada aluno apresente as suas habilidades.

Em turmas de colégios isso seria bastante diferente, pois a interação entre os alunos ocorre de maneira diferente, os currículos variam de um colégio para outro etc.. Nesse caso, o tutor funciona melhor como uma ferramenta de suporte às atividades de aprendizagem, onde depois de terem sido expostos a uma aula, cada aluno pode resolver o seu conjunto de exercícios de maneira individual e os professores ganham flexibilidade para dar mais atenção aos alunos com dificuldades. Mas no início é necessário investir um tempo na familiarização com a ferramenta.

PROPOSTA DE TRABALHO FUTURO

Sistemas Tutores Cognitivos é uma das linhas de pesquisa dentro da área de IA-ED. Em trabalhos futuros pretende-se explorar, dentro dessa linha a utilização de técnicas de IA para Planejamento e Diagnóstico (Koedinger & Anderson, 1990) e (Koedinger & Anderson, 1993), (Koning et al., 2000).

BIBLIOGRAFIA

ACT-R Research Group (2002) http://act-r.psy.cmu.edu/.

Aleven, V., & Koedinger, K.R. (2000). Limitations of Student Control: Do Students Know When They Need Help? In G. Gauthier, C. Frasson, and K. VanLehn (Eds.), Proceedings of the 5th International Conference on Intelligent Tutoring Systems, ITS 2000 (pp. 292-303). Berlin: Springer Verlag. Best Paper Award ITS 2000.

Aleven, V., & Koedinger, K.R. (2000). The Need For Tutorial Dialog To Support Self-Explanation. In C.P. Rose & R. Freedman (Eds.), Building Dialogue Systems for Tutorial Applications, Papers of the 2000 AAAI Fall Symposium (pp. 65-73). Technical Report FS-00-01. Menlo Park, CA: AAAI Press.

Anderson, J. R. (1976). *Language, memory, and thought*. Hillsdale, NJ: Erlbaum.

Anderson, J. R. (1983). *The Architecture of Cognition*. Cambridge, MA: Harvard University Press

Anderson, J. R. (1983a). Knowledge compilation: The general learning mechanisms. In *Proceedings of the 1983 Machine Learning Workshop*, 203-212. Anderson, J. R., & Kosslyn, S. M. (Eds.), (1984). *Essays on Learning and Memory.* San Francisco, CA: Freeman.

Anderson, J. R., Boyle, C. F., & Reiser, B. J. (1985). Intelligent Tutoring Systems. Science, 228, 456-467.

Anderson, J. R., & Reiser, B. J. (1985a). The LISP tutor. *Byte*, 10, 159-175.

Anderson, J. R., Boyle, C. F., & Yost, G. (1985b). The geometry tutor. In *Proceedings of IJCAI*, 1-7.

Anderson, J. R. (1986). Knowledge compilation: The general learning mechanism. In R. Michalski, J. Carbonell, & T. Mitchell (Eds.), *Machine learning II*. Los Altos, CA: Morgan Kaufmann.

Anderson, J. R., Boyle, C. F., Farrell, R., & Reiser, B. J. (1987). Cognitive principles in the design of computer tutors. In P. Morris (Ed.), *Modelling Cognition*, Wiley.

Anderson, J. R. (1989). A theory of human knowledge. *Artificial Intelligence*, 40, 313-351.

Anderson, J. R., Boyle, C. F., Corbett, A., and Lewis, M. W. (1990) Cognitive modelling and intelligent tutoring. *Artificial Intelligence*, 42, 7-49.

Anderson, J. R. (1993). Rules of the Mind. Hillsdale, NJ: Erlbaum.

Anderson, J.R., Corbett, A.T., Koedinger, K.R., & Pelletier, R. (1995). Cognitive Tutors: Lessons Learned. The Journal of the Learning Sciences, 4 (2), 167-207.

Anderson, J. R. & Lebiere, C. (1998). *The atomic components of thought.* Mahwah, NJ: Erlbaum. [Chapter abstracts, model source code, and web-based simulations]

Bunderson, C. V., & Faust, G. W. (1976) Programmed and computer-assisted instruction. In N. L. Gage (Ed.), The psychology of teaching Methods: 75th Yearbook of the National Society for the study of Education (pp. xxx-xxx). Chicago: University of Chicago Press.

Corbett, A.T. and Anderson, J.R. (1995). Knowledge decomposition and subgoal reification in the ACT programming tutor. Artificial Intelligence and Education, 1995: The Proceedings of AI-ED 95. Charlottesville, VA: AACE.

Corbett, A. T., Koedinger, K. R., & Anderson, J. R. (1997). Intelligent tutoring systems (Chapter 37). M. G. Helander, T. K. Landauer, & P. Prabhu, (Eds.) Handbook of Human-Computer Interaction, 2nd edition. Amsterdam, The Netherlands: Elsevier Science.

Gagné, R. & Briggs, L. J., (1974) Principles of Instructional Design. New York: Holt, Rinehart & Winston.

Koedinger, K. R., & Anderson, J. R. (1990). Abstract planning and perceptual chunks: Elements of expertise in geometry. *Cognitive Science*, 14, 511-550.

Koedinger, K. & Anderson, J. R. (1993). Reifying implicit planning in geometry: Guidelines for model-based intelligent tutoring system design.In S. P. Lajoie & S. J. Derry (Eds.) Computers as cognitive tools (pp. 15-46). Hillsdale, NJ: Erlbaum.

Koedinger, K. R. (1998). Intellige nt cognitive tutors as modeling tool and instructional model. Invited paper for the National Council of Teachers of Mathematics Standards 2000 Technology Conference

Koning, K. de, Bredeweg, B., Breuker, J. & B. Wielinga. (2000). Model-Based Reasoning about Learner Behaviour. Artificial Intelligence, Volume 117, Number 2, pages 173-229.

Lewis, M. W., & Anderson, J. R. (1985). Discrimination of operator schemata in problem solving: Procedural learning from examples. *Cognitive Psychology*, 17, 26-65.

Nilsson, N. J. (1982). Principles of Artificial Intelligence. Springer-Verlag, New York.

Russel, S. J. & Norvig, P. (2002) Artificial Intelligence: A Modern Approach (2nd Edition) - Prentice Hall series in Artificial Intelligence.

Self, J., Computational Mathetics. http://www.cbl.leeds.ac.uk/~jas/cm.html.

Taatgen, N., (1999) Learning without limits - From Problem Solving towards a Unified Theory of Learning - Phd Thesis.

Wenger, E. (1987). Artificial Intelligence and Tutoring Systems, Los Altos: Morgan Kaufmann.

Winograd, T., (1975) Frame representation and the declarative procedural controversy, in D. Bobrow and A. Collins (Eds.), Representation and Understanding (Academic Press, New York,).