CLIPS 6.4 - Guia do Usuário (Traduzido)

Guia do Usuário do CLIPS 6.4

Joseph C. Giarratano, Ph.D.

LEIA-ME

O primeiro passo rumo à sabedoria é admitir a ignorância.
O segundo passo é perceber que você não precisa contar isso ao mundo.

Esta seção era chamada de “Prefácio”, mas como ninguém lia, foi renomeada para um título que usuários de computador estão condicionados a obedecer. Uma sugestão alternativa era chamá-la de “Não-leia-me”, mas como as pessoas hoje acreditam em tudo o que leem, temi que realmente não lessem.

O objetivo de um Prefácio – opa, desculpe, de um Leia-me – é fornecer metaconhecimento sobre o conhecimento contido no livro. Metaconhecimento é conhecimento sobre o conhecimento. Então, esta descrição do Leia-me é, na verdade, um metametaconhecimento. Se você está confuso ou intrigado até aqui, siga em frente e leia este livro de qualquer maneira – preciso de todos os leitores que puder conseguir.

O QUE É O CLIPS?

CLIPS é uma ferramenta para desenvolvimento de sistemas especialistas, originalmente desenvolvida pela Software Technology Branch da NASA no Centro Espacial Lyndon B. Johnson. Desde seu primeiro lançamento em 1986, CLIPS passou por refinamentos

contínuos, sendo usado atualmente por milhares de pessoas em todo o mundo.

CLIPS é projetado para facilitar o desenvolvimento de software que modele conhecimento ou especialização humana.

O conhecimento pode ser representado no CLIPS de três formas:

- Regras, usadas principalmente para conhecimento heurístico baseado em experiência;
- Defunções e funções genéricas, voltadas para conhecimento procedural;
- Programação orientada a objetos, também voltada para conhecimento procedural, suportando as cinco características da OOP: classes, manipuladores de mensagens, abstração, encapsulamento, herança e polimorfismo.

Você pode desenvolver software usando apenas regras, apenas objetos, ou uma combinação dos dois.

CLIPS também foi projetado para integração com outras linguagens como C e Java. O nome CLIPS é uma sigla para C Language Integrated Production System. Regras e objetos formam um sistema integrado, já que regras podem casar padrões com fatos e objetos.

Você pode usar o CLIPS como ferramenta autônoma ou integrá-lo com outras linguagens. Da mesma forma, código procedural pode ser definido como funções externas e chamado a partir do CLIPS.

CLIPS 6.4 - Guia do Usuário (Traduzido)

Se você já está familiarizado com programação orientada a objetos em outras linguagens, saberá os benefícios. Caso não esteja, CLIPS é uma ótima ferramenta para aprender esse conceito moderno de desenvolvimento de software.

SOBRE ESTE LIVRO

Este guia é um tutorial introdutório sobre os recursos básicos do CLIPS. Para uma discussão completa, consulte o Manual de Referência do CLIPS.

PARA QUEM É ESTE LIVRO

Este guia é para quem tem pouca ou nenhuma experiência com sistemas especialistas. Pode ser usado em sala de aula ou para autoestudo. É necessário apenas conhecimento básico de programação em linguagens como Java, C, Ada, etc.

COMO USAR ESTE LIVRO

Este livro é voltado para quem quer uma introdução prática e rápida. Os exemplos são simples e o estilo de escrita é leve e com humor. Para aproveitar melhor, digite os exemplos e veja o funcionamento real. Compare os resultados e leia também o Manual de Referência à medida que avança nos capítulos.

CAPÍTULO 1 – APENAS OS FATOS

CLIPS 6.4 - Guia do Usuário (Traduzido)

“Se você ignorar os fatos, nunca se preocupará em estar errado.”

CLIPS é uma linguagem voltada para escrever sistemas especialistas, que são programas que modelam o conhecimento humano. Diferente de programas convencionais como editores de texto ou planilhas, os sistemas especialistas lidam com conhecimento e inferência.

CLIPS é chamado de ferramenta para sistemas especialistas porque é um ambiente completo com editor integrado e ferramentas de depuração. A parte que executa inferências é chamada de shell do CLIPS.

Os componentes principais de um sistema especialista CLIPS são:

1. Lista de fatos e instâncias: memória global de dados.
2. Base de conhecimento: contém todas as regras (base de regras).
3. Mecanismo de inferência: controla a execução das regras.

Os programas CLIPS consistem em fatos, regras e objetos. O motor de inferência determina quais regras devem ser executadas. Isso diferencia o CLIPS de linguagens procedurais, nas quais a execução ocorre mesmo sem dados explícitos.

(...continua nos próximos capítulos...)

CLIPS 6.4 - Guia do Usuário (Traduzido)

Guia do Usuário do CLIPS 6.4

Joseph C. Giarratano, Ph.D.

LEIA-ME

O primeiro passo rumo à sabedoria é admitir a ignorância.
O segundo passo é perceber que você não precisa contar isso ao mundo.

Esta seção era chamada de “Prefácio”, mas como ninguém lia, foi renomeada para um título que usuários de computador estão condicionados a obedecer. Uma sugestão alternativa era chamá-la de “Não-leia-me”, mas como as pessoas hoje acreditam em tudo o que leem, temi que realmente não lessem.

O objetivo de um Prefácio – opa, desculpe, de um Leia-me – é fornecer metaconhecimento sobre o conhecimento contido no livro. Metaconhecimento é conhecimento sobre o conhecimento. Então, esta descrição do Leia-me é, na verdade, um metametaconhecimento. Se você está confuso ou intrigado até aqui, siga em frente e leia este livro de qualquer maneira – preciso de todos os leitores que puder conseguir.

O QUE É O CLIPS?

CLIPS é uma ferramenta para desenvolvimento de sistemas especialistas, originalmente desenvolvida pela Software Technology Branch da NASA no Centro Espacial Lyndon B. Johnson. Desde seu primeiro lançamento em 1986, CLIPS passou por refinamentos

contínuos, sendo usado atualmente por milhares de pessoas em todo o mundo.

CLIPS é projetado para facilitar o desenvolvimento de software que modele conhecimento ou especialização humana.

O conhecimento pode ser representado no CLIPS de três formas:

- Regras, usadas principalmente para conhecimento heurístico baseado em experiência;
- Defunções e funções genéricas, voltadas para conhecimento procedural;
- Programação orientada a objetos, também voltada para conhecimento procedural, suportando as cinco características da OOP: classes, manipuladores de mensagens, abstração, encapsulamento, herança e polimorfismo.

Você pode desenvolver software usando apenas regras, apenas objetos, ou uma combinação dos dois.

CLIPS também foi projetado para integração com outras linguagens como C e Java. O nome CLIPS é uma sigla para C Language Integrated Production System. Regras e objetos formam um sistema integrado, já que regras podem casar padrões com fatos e objetos.

Você pode usar o CLIPS como ferramenta autônoma ou integrá-lo com outras linguagens. Da mesma forma, código procedural pode ser definido como funções externas e chamado a partir do CLIPS.

CLIPS 6.4 - Guia do Usuário (Traduzido)

Se você já está familiarizado com programação orientada a objetos em outras linguagens, saberá os benefícios. Caso não esteja, CLIPS é uma ótima ferramenta para aprender esse conceito moderno de desenvolvimento de software.

SOBRE ESTE LIVRO

Este guia é um tutorial introdutório sobre os recursos básicos do CLIPS. Para uma discussão completa, consulte o Manual de Referência do CLIPS.

PARA QUEM É ESTE LIVRO

Este guia é para quem tem pouca ou nenhuma experiência com sistemas especialistas. Pode ser usado em sala de aula ou para autoestudo. É necessário apenas conhecimento básico de programação em linguagens como Java, C, Ada, etc.

COMO USAR ESTE LIVRO

Este livro é voltado para quem quer uma introdução prática e rápida. Os exemplos são simples e o estilo de escrita é leve e com humor. Para aproveitar melhor, digite os exemplos e veja o funcionamento real. Compare os resultados e leia também o Manual de Referência à medida que avança nos capítulos.

CAPÍTULO 1 – APENAS OS FATOS

CLIPS 6.4 - Guia do Usuário (Traduzido)

“Se você ignorar os fatos, nunca se preocupará em estar errado.”

CLIPS é uma linguagem voltada para escrever sistemas especialistas, que são programas que modelam o conhecimento humano. Diferente de programas convencionais como editores de texto ou planilhas, os sistemas especialistas lidam com conhecimento e inferência.

CLIPS é chamado de ferramenta para sistemas especialistas porque é um ambiente completo com editor integrado e ferramentas de depuração. A parte que executa inferências é chamada de shell do CLIPS.

Os componentes principais de um sistema especialista CLIPS são:

1. Lista de fatos e instâncias: memória global de dados.
2. Base de conhecimento: contém todas as regras (base de regras).
3. Mecanismo de inferência: controla a execução das regras.

Os programas CLIPS consistem em fatos, regras e objetos. O motor de inferência determina quais regras devem ser executadas. Isso diferencia o CLIPS de linguagens procedurais, nas quais a execução ocorre mesmo sem dados explícitos.

(...continua nos próximos capítulos...)

CLIPS 6.4 - Guia do Usuário (Traduzido)

CAPÍTULO 2 – SEGUINDO AS REGRAS

“Se você quer chegar a algum lugar na vida, não quebre as regras — crie-as!”

FAZENDO BOAS REGRAS

Para realizar tarefas úteis, um sistema especialista precisa de regras além de fatos. Já vimos como os fatos são inseridos e removidos, agora veremos como funcionam as regras. Uma regra é semelhante a uma instrução IF THEN em linguagens como Java, C ou Ada.

Exemplo em pseudocódigo:
SE certas condições forem verdadeiras
ENTÃO execute as seguintes ações

A tradução disso para CLIPS é simples. Um exemplo de regra que trata sons de patos pode ser:

SE o animal é um pato
ENTÃO o som emitido é "quack"

Em CLIPS, isso seria representado como:

(defrule pato

```
   (animal-is duck)
=>
   (assert (sound-is quack)))
```

A palavra-chave `defrule` define a regra e o nome da regra vem logo após. A parte antes do símbolo `=>` é chamada de LHS (lado esquerdo) e representa as condições. A parte após `=>` é o RHS (lado direito), ou as ações que ocorrem se as condições forem verdadeiras.

Comentários podem ser adicionados com ponto e vírgula `;` ou usando uma string logo após o nome da regra:

```
(defrule pato
   "Aqui vem o quack"
   (animal-is duck)
=>
   (assert (sound-is quack)))
```

Somente um nome pode ser usado por regra, então uma nova definição com o mesmo nome substitui a anterior. Após definir uma regra, ela será ativada quando os fatos no lado esquerdo corresponderem aos fatos da lista.

A execução ocorre com o comando (run). Se os padrões corresponderem, a regra é ativada e adicionada à agenda. A agenda contém as ativações – regras prontas para disparar. As ativações são ordenadas por prioridade (salience).

Você pode ver o que está na agenda com o comando:

(agenda)

Quando (run) é executado, CLIPS dispara a regra com maior salience, executa suas ações, remove a ativação e repete até que a agenda esteja vazia.

CLIPS impede que a mesma regra seja ativada novamente sobre o mesmo fato. Isso se chama refração. Para disparar a mesma regra novamente, é preciso retirar (retract) o fato e inseri-lo novamente (assert).

Você pode ver o conteúdo de uma regra com:

(ppdefrule nome-da-regra)

E listar todas as regras com:

(rules)

SAÍDA FORMATADA

Você pode usar (printout) para imprimir informações na tela:

(defrule pato

```
  (animal-is duck)
=>
  (printout t "quack" crlf))
```

`t` indica que a saída deve ir para o terminal padrão. `crlf` adiciona quebra de linha.

PRIORIDADES E CARREGAMENTO

Regras podem ter prioridade com `declare (salience <valor>)`. Regras com maior salience disparam antes. Você pode salvar regras com:

```
(save "arquivo.clp")
```

E recarregá-las com:

```
(load "arquivo.clp")
```

Arquivos binários podem ser salvos com (bsave) e carregados com (bload).

Você também pode salvar e carregar fatos com:

```
(save-facts "fatos.dat")
(load-facts "fatos.dat")
```

O comando (batch "arquivo") executa comandos de um arquivo como se fossem

digitados.

DEBUG E OBSERVAÇÃO

Use (watch facts), (watch rules), (watch activations) para depurar.

Use (run 1) para executar uma regra por vez.

Também é possível configurar breakpoints com:

(set-break nome-da-regra)
(remove-break nome-da-regra)
(show-breaks)

Esses são os fundamentos do uso de regras no CLIPS. No próximo capítulo, abordaremos como detalhar mais os programas com múltiplas regras.

CLIPS 6.4 - Guia do Usuário (Traduzido)

Guia do Usuário do CLIPS 6.4

Joseph C. Giarratano, Ph.D.

LEIA-ME

O primeiro passo rumo à sabedoria é admitir a ignorância.
O segundo passo é perceber que você não precisa contar isso ao mundo.

Esta seção era chamada de “Prefácio”, mas como ninguém lia, foi renomeada para um título que usuários de computador estão condicionados a obedecer. Uma sugestão alternativa era chamá-la de “Não-leia-me”, mas como as pessoas hoje acreditam em tudo o que leem, temi que realmente não lessem.

O objetivo de um Prefácio – opa, desculpe, de um Leia-me – é fornecer metaconhecimento sobre o conhecimento contido no livro. Metaconhecimento é conhecimento sobre o conhecimento. Então, esta descrição do Leia-me é, na verdade, um metametaconhecimento. Se você está confuso ou intrigado até aqui, siga em frente e leia este livro de qualquer maneira – preciso de todos os leitores que puder conseguir.

O QUE É O CLIPS?

CLIPS é uma ferramenta para desenvolvimento de sistemas especialistas, originalmente desenvolvida pela Software Technology Branch da NASA no Centro Espacial Lyndon B. Johnson. Desde seu primeiro lançamento em 1986, CLIPS passou por refinamentos

contínuos, sendo usado atualmente por milhares de pessoas em todo o mundo.

CLIPS é projetado para facilitar o desenvolvimento de software que modele conhecimento ou especialização humana.

O conhecimento pode ser representado no CLIPS de três formas:

- Regras, usadas principalmente para conhecimento heurístico baseado em experiência;
- Defunções e funções genéricas, voltadas para conhecimento procedural;
- Programação orientada a objetos, também voltada para conhecimento procedural, suportando as cinco características da OOP: classes, manipuladores de mensagens, abstração, encapsulamento, herança e polimorfismo.

Você pode desenvolver software usando apenas regras, apenas objetos, ou uma combinação dos dois.

CLIPS também foi projetado para integração com outras linguagens como C e Java. O nome CLIPS é uma sigla para C Language Integrated Production System. Regras e objetos formam um sistema integrado, já que regras podem casar padrões com fatos e objetos.

Você pode usar o CLIPS como ferramenta autônoma ou integrá-lo com outras linguagens. Da mesma forma, código procedural pode ser definido como funções externas e chamado a partir do CLIPS.

CLIPS 6.4 - Guia do Usuário (Traduzido)

Se você já está familiarizado com programação orientada a objetos em outras linguagens, saberá os benefícios. Caso não esteja, CLIPS é uma ótima ferramenta para aprender esse conceito moderno de desenvolvimento de software.

SOBRE ESTE LIVRO

Este guia é um tutorial introdutório sobre os recursos básicos do CLIPS. Para uma discussão completa, consulte o Manual de Referência do CLIPS.

PARA QUEM É ESTE LIVRO

Este guia é para quem tem pouca ou nenhuma experiência com sistemas especialistas. Pode ser usado em sala de aula ou para autoestudo. É necessário apenas conhecimento básico de programação em linguagens como Java, C, Ada, etc.

COMO USAR ESTE LIVRO

Este livro é voltado para quem quer uma introdução prática e rápida. Os exemplos são simples e o estilo de escrita é leve e com humor. Para aproveitar melhor, digite os exemplos e veja o funcionamento real. Compare os resultados e leia também o Manual de Referência à medida que avança nos capítulos.

CAPÍTULO 1 – APENAS OS FATOS

CLIPS 6.4 - Guia do Usuário (Traduzido)

“Se você ignorar os fatos, nunca se preocupará em estar errado.”

CLIPS é uma linguagem voltada para escrever sistemas especialistas, que são programas que modelam o conhecimento humano. Diferente de programas convencionais como editores de texto ou planilhas, os sistemas especialistas lidam com conhecimento e inferência.

CLIPS é chamado de ferramenta para sistemas especialistas porque é um ambiente completo com editor integrado e ferramentas de depuração. A parte que executa inferências é chamada de shell do CLIPS.

Os componentes principais de um sistema especialista CLIPS são:

1. Lista de fatos e instâncias: memória global de dados.
2. Base de conhecimento: contém todas as regras (base de regras).
3. Mecanismo de inferência: controla a execução das regras.

Os programas CLIPS consistem em fatos, regras e objetos. O motor de inferência determina quais regras devem ser executadas. Isso diferencia o CLIPS de linguagens procedurais, nas quais a execução ocorre mesmo sem dados explícitos.

(...continua nos próximos capítulos...)

CLIPS 6.4 - Guia do Usuário (Traduzido)

CAPÍTULO 2 – SEGUINDO AS REGRAS

“Se você quer chegar a algum lugar na vida, não quebre as regras — crie-as!”

FAZENDO BOAS REGRAS

Para realizar tarefas úteis, um sistema especialista precisa de regras além de fatos. Já vimos como os fatos são inseridos e removidos, agora veremos como funcionam as regras. Uma regra é semelhante a uma instrução IF THEN em linguagens como Java, C ou Ada.

Exemplo em pseudocódigo:
SE certas condições forem verdadeiras
ENTÃO execute as seguintes ações

A tradução disso para CLIPS é simples. Um exemplo de regra que trata sons de patos pode ser:

SE o animal é um pato
ENTÃO o som emitido é "quack"

Em CLIPS, isso seria representado como:

(defrule pato

```
   (animal-is duck)
=>
   (assert (sound-is quack)))
```

A palavra-chave `defrule` define a regra e o nome da regra vem logo após. A parte antes do símbolo `=>` é chamada de LHS (lado esquerdo) e representa as condições. A parte após `=>` é o RHS (lado direito), ou as ações que ocorrem se as condições forem verdadeiras.

Comentários podem ser adicionados com ponto e vírgula `;` ou usando uma string logo após o nome da regra:

```
(defrule pato
   "Aqui vem o quack"
   (animal-is duck)
=>
   (assert (sound-is quack)))
```

Somente um nome pode ser usado por regra, então uma nova definição com o mesmo nome substitui a anterior. Após definir uma regra, ela será ativada quando os fatos no lado esquerdo corresponderem aos fatos da lista.

A execução ocorre com o comando (run). Se os padrões corresponderem, a regra é ativada e adicionada à agenda. A agenda contém as ativações – regras prontas para disparar. As ativações são ordenadas por prioridade (salience).

Você pode ver o que está na agenda com o comando:

(agenda)

Quando (run) é executado, CLIPS dispara a regra com maior salience, executa suas ações, remove a ativação e repete até que a agenda esteja vazia.

CLIPS impede que a mesma regra seja ativada novamente sobre o mesmo fato. Isso se chama refração. Para disparar a mesma regra novamente, é preciso retirar (retract) o fato e inseri-lo novamente (assert).

Você pode ver o conteúdo de uma regra com:

(ppdefrule nome-da-regra)

E listar todas as regras com:

(rules)

SAÍDA FORMATADA

Você pode usar (printout) para imprimir informações na tela:

(defrule pato

```
  (animal-is duck)
=>
  (printout t "quack" crlf))
```

`t` indica que a saída deve ir para o terminal padrão. `crlf` adiciona quebra de linha.

PRIORIDADES E CARREGAMENTO

Regras podem ter prioridade com `declare (salience <valor>)`. Regras com maior salience disparam antes. Você pode salvar regras com:

```
(save "arquivo.clp")
```

E recarregá-las com:

```
(load "arquivo.clp")
```

Arquivos binários podem ser salvos com (bsave) e carregados com (bload).

Você também pode salvar e carregar fatos com:

```
(save-facts "fatos.dat")
(load-facts "fatos.dat")
```

O comando (batch "arquivo") executa comandos de um arquivo como se fossem

digitados.

DEBUG E OBSERVAÇÃO

Use (watch facts), (watch rules), (watch activations) para depurar.

Use (run 1) para executar uma regra por vez.

Também é possível configurar breakpoints com:

(set-break nome-da-regra)
(remove-break nome-da-regra)
(show-breaks)

Esses são os fundamentos do uso de regras no CLIPS. No próximo capítulo, abordaremos como detalhar mais os programas com múltiplas regras.

CAPÍTULO 3 – ADICIONANDO DETALHES

“Não é a visão geral que causa problemas — são os detalhes.”

Nos dois primeiros capítulos, você aprendeu os fundamentos do CLIPS. Agora veremos como construir sobre essa base para criar programas mais poderosos.

PARE E SIGA

Até agora, vimos o exemplo mais simples: um programa com uma única regra. Mas sistemas especialistas reais precisam de múltiplas regras. Vamos examinar um exemplo prático: um robô móvel que deve responder a um semáforo.

Podemos usar regras como:

```
(defrule luz-vermelha
   (luz vermelha)
=>
   (printout t "Pare" crlf))

(defrule luz-verde
   (luz verde)
=>
   (printout t "Siga" crlf))
```

Ao afirmar (luz vermelha), o robô imprime "Pare". Ao afirmar (luz verde), imprime "Siga".

DANDO UMA VOLTA

Mas há mais possibilidades: luzes com seta verde, sinais de “andar” e “não andar”, mãos vermelhas para pedestres. Assim, se o robô estiver andando, ele deve prestar atenção nesses sinais.

Podemos representar essa lógica com mais de um padrão por regra:

```
(defrule atravesse
   (status andando)
   (sinal-andar andar)
=>
   (printout t "Siga" crlf))
```

A regra só é ativada quando ambos os padrões estão presentes. Isso é um exemplo de **condição lógica AND** — todos os padrões devem ser verdadeiros para que a regra seja ativada.

ESTRATÉGIA DE EXECUÇÃO

CLIPS usa uma **estratégia de resolução de conflitos** para decidir qual regra ativar

quando múltiplas estão prontas. As estratégias disponíveis são: `depth`, `breadth`, `LEX`, `MEA`, `complexity`, `simplicity` e `random`.

A estratégia padrão é `depth`, na qual ativações mais recentes com mesma prioridade são executadas antes das mais antigas. Essa estratégia pode ser modificada, mas todos os exemplos neste guia assumem `depth`.

DICA: se você executar um sistema de outra pessoa, assegure-se de usar as mesmas configurações. O ideal é que as configurações sejam codificadas explicitamente no sistema.

USANDO DEFFACTS

Se estiver cansado de digitar os mesmos fatos, use `deffacts` para pré-carregá-los:

```
(deffacts andar "Fatos sobre andar"
   (status andando)
   (sinal-andar andar))
```

Esses fatos são inseridos com o comando (reset), que:

1. Remove fatos antigos e ativações.
2. Insere fatos definidos com `deffacts`.

Com isso, você pode limpar o ambiente sem apagar suas regras.

REMOVENDO DEFFACTS

Use `undeffacts` para remover um conjunto de fatos definidos:

(undeffacts andar)
(reset)

Isso evita que os fatos sejam reassertados. Para restaurar, você deve redefinir o `deffacts`.

MONITORANDO O SISTEMA

Você pode observar regras e ativações com:

(watch rules)
(watch activations)
(watch statistics)

Estatísticas incluem: tempo de execução, número de regras ativadas, média e máximo de fatos/ativações.

Use o comando:

(dribble-on "log.txt")

para registrar todas as entradas e saídas. Para parar:

(dribble-off)

O comando (run N) executa apenas N ativações, útil para depuração passo a passo.

PONTOS DE PARADA

Para depurar regras específicas, use breakpoints:

(set-break nome-da-regra)
(remove-break nome-da-regra)
(show-breaks)

CASAMENTO DE PADRÕES

Se uma regra não está ativando como esperado, use o comando (matches nome-da-regra) para verificar quais padrões casam com os fatos atuais.

Ele informará quais padrões são compatíveis, quais não são e se a regra está ativada.

Por exemplo:

(defrule ferias

```
   (trabalho feito)
   (dinheiro muito)
   (reservas feitas)
=>
   (printout t "Vamos viajar!" crlf))
```

```
(matches ferias)
```

...informará quais partes estão sendo satisfeitas pelos fatos atuais.

No próximo capítulo, vamos explorar o uso de variáveis para tornar os padrões ainda mais dinâmicos.

CLIPS 6.4 - Guia do Usuário (Traduzido)

Guia do Usuário do CLIPS 6.4

Joseph C. Giarratano, Ph.D.

LEIA-ME

O primeiro passo rumo à sabedoria é admitir a ignorância.
O segundo passo é perceber que você não precisa contar isso ao mundo.

Esta seção era chamada de “Prefácio”, mas como ninguém lia, foi renomeada para um título que usuários de computador estão condicionados a obedecer. Uma sugestão alternativa era chamá-la de “Não-leia-me”, mas como as pessoas hoje acreditam em tudo o que leem, temi que realmente não lessem.

O objetivo de um Prefácio – opa, desculpe, de um Leia-me – é fornecer metaconhecimento sobre o conhecimento contido no livro. Metaconhecimento é conhecimento sobre o conhecimento. Então, esta descrição do Leia-me é, na verdade, um metametaconhecimento. Se você está confuso ou intrigado até aqui, siga em frente e leia este livro de qualquer maneira – preciso de todos os leitores que puder conseguir.

O QUE É O CLIPS?

CLIPS é uma ferramenta para desenvolvimento de sistemas especialistas, originalmente desenvolvida pela Software Technology Branch da NASA no Centro Espacial Lyndon B. Johnson. Desde seu primeiro lançamento em 1986, CLIPS passou por refinamentos

contínuos, sendo usado atualmente por milhares de pessoas em todo o mundo.

CLIPS é projetado para facilitar o desenvolvimento de software que modele conhecimento ou especialização humana.

O conhecimento pode ser representado no CLIPS de três formas:

- Regras, usadas principalmente para conhecimento heurístico baseado em experiência;
- Defunções e funções genéricas, voltadas para conhecimento procedural;
- Programação orientada a objetos, também voltada para conhecimento procedural, suportando as cinco características da OOP: classes, manipuladores de mensagens, abstração, encapsulamento, herança e polimorfismo.

Você pode desenvolver software usando apenas regras, apenas objetos, ou uma combinação dos dois.

CLIPS também foi projetado para integração com outras linguagens como C e Java. O nome CLIPS é uma sigla para C Language Integrated Production System. Regras e objetos formam um sistema integrado, já que regras podem casar padrões com fatos e objetos.

Você pode usar o CLIPS como ferramenta autônoma ou integrá-lo com outras linguagens. Da mesma forma, código procedural pode ser definido como funções externas e chamado a partir do CLIPS.

CLIPS 6.4 - Guia do Usuário (Traduzido)

Se você já está familiarizado com programação orientada a objetos em outras linguagens, saberá os benefícios. Caso não esteja, CLIPS é uma ótima ferramenta para aprender esse conceito moderno de desenvolvimento de software.

SOBRE ESTE LIVRO

Este guia é um tutorial introdutório sobre os recursos básicos do CLIPS. Para uma discussão completa, consulte o Manual de Referência do CLIPS.

PARA QUEM É ESTE LIVRO

Este guia é para quem tem pouca ou nenhuma experiência com sistemas especialistas. Pode ser usado em sala de aula ou para autoestudo. É necessário apenas conhecimento básico de programação em linguagens como Java, C, Ada, etc.

COMO USAR ESTE LIVRO

Este livro é voltado para quem quer uma introdução prática e rápida. Os exemplos são simples e o estilo de escrita é leve e com humor. Para aproveitar melhor, digite os exemplos e veja o funcionamento real. Compare os resultados e leia também o Manual de Referência à medida que avança nos capítulos.

CAPÍTULO 1 – APENAS OS FATOS

CLIPS 6.4 - Guia do Usuário (Traduzido)

“Se você ignorar os fatos, nunca se preocupará em estar errado.”

CLIPS é uma linguagem voltada para escrever sistemas especialistas, que são programas que modelam o conhecimento humano. Diferente de programas convencionais como editores de texto ou planilhas, os sistemas especialistas lidam com conhecimento e inferência.

CLIPS é chamado de ferramenta para sistemas especialistas porque é um ambiente completo com editor integrado e ferramentas de depuração. A parte que executa inferências é chamada de shell do CLIPS.

Os componentes principais de um sistema especialista CLIPS são:

1. Lista de fatos e instâncias: memória global de dados.
2. Base de conhecimento: contém todas as regras (base de regras).
3. Mecanismo de inferência: controla a execução das regras.

Os programas CLIPS consistem em fatos, regras e objetos. O motor de inferência determina quais regras devem ser executadas. Isso diferencia o CLIPS de linguagens procedurais, nas quais a execução ocorre mesmo sem dados explícitos.

(...continua nos próximos capítulos...)

CLIPS 6.4 - Guia do Usuário (Traduzido)

CAPÍTULO 2 – SEGUINDO AS REGRAS

“Se você quer chegar a algum lugar na vida, não quebre as regras — crie-as!”

FAZENDO BOAS REGRAS

Para realizar tarefas úteis, um sistema especialista precisa de regras além de fatos. Já vimos como os fatos são inseridos e removidos, agora veremos como funcionam as regras. Uma regra é semelhante a uma instrução IF THEN em linguagens como Java, C ou Ada.

Exemplo em pseudocódigo:
SE certas condições forem verdadeiras
ENTÃO execute as seguintes ações

A tradução disso para CLIPS é simples. Um exemplo de regra que trata sons de patos pode ser:

SE o animal é um pato
ENTÃO o som emitido é "quack"

Em CLIPS, isso seria representado como:

(defrule pato

```
   (animal-is duck)
=>
   (assert (sound-is quack)))
```

A palavra-chave `defrule` define a regra e o nome da regra vem logo após. A parte antes do símbolo `=>` é chamada de LHS (lado esquerdo) e representa as condições. A parte após `=>` é o RHS (lado direito), ou as ações que ocorrem se as condições forem verdadeiras.

Comentários podem ser adicionados com ponto e vírgula `;` ou usando uma string logo após o nome da regra:

```
(defrule pato
   "Aqui vem o quack"
   (animal-is duck)
=>
   (assert (sound-is quack)))
```

Somente um nome pode ser usado por regra, então uma nova definição com o mesmo nome substitui a anterior. Após definir uma regra, ela será ativada quando os fatos no lado esquerdo corresponderem aos fatos da lista.

A execução ocorre com o comando (run). Se os padrões corresponderem, a regra é ativada e adicionada à agenda. A agenda contém as ativações – regras prontas para disparar. As ativações são ordenadas por prioridade (salience).

Você pode ver o que está na agenda com o comando:

(agenda)

Quando (run) é executado, CLIPS dispara a regra com maior salience, executa suas ações, remove a ativação e repete até que a agenda esteja vazia.

CLIPS impede que a mesma regra seja ativada novamente sobre o mesmo fato. Isso se chama refração. Para disparar a mesma regra novamente, é preciso retirar (retract) o fato e inseri-lo novamente (assert).

Você pode ver o conteúdo de uma regra com:

(ppdefrule nome-da-regra)

E listar todas as regras com:

(rules)

SAÍDA FORMATADA

Você pode usar (printout) para imprimir informações na tela:

(defrule pato

```
  (animal-is duck)
=>
  (printout t "quack" crlf))
```

`t` indica que a saída deve ir para o terminal padrão. `crlf` adiciona quebra de linha.

PRIORIDADES E CARREGAMENTO

Regras podem ter prioridade com `declare (salience <valor>)`. Regras com maior salience disparam antes. Você pode salvar regras com:

```
(save "arquivo.clp")
```

E recarregá-las com:

```
(load "arquivo.clp")
```

Arquivos binários podem ser salvos com (bsave) e carregados com (bload).

Você também pode salvar e carregar fatos com:

```
(save-facts "fatos.dat")
(load-facts "fatos.dat")
```

O comando (batch "arquivo") executa comandos de um arquivo como se fossem

digitados.

DEBUG E OBSERVAÇÃO

Use (watch facts), (watch rules), (watch activations) para depurar.

Use (run 1) para executar uma regra por vez.

Também é possível configurar breakpoints com:

(set-break nome-da-regra)
(remove-break nome-da-regra)
(show-breaks)

Esses são os fundamentos do uso de regras no CLIPS. No próximo capítulo, abordaremos como detalhar mais os programas com múltiplas regras.

CAPÍTULO 3 – ADICIONANDO DETALHES

“Não é a visão geral que causa problemas — são os detalhes.”

Nos dois primeiros capítulos, você aprendeu os fundamentos do CLIPS. Agora veremos como construir sobre essa base para criar programas mais poderosos.

PARE E SIGA

Até agora, vimos o exemplo mais simples: um programa com uma única regra. Mas sistemas especialistas reais precisam de múltiplas regras. Vamos examinar um exemplo prático: um robô móvel que deve responder a um semáforo.

Podemos usar regras como:

```
(defrule luz-vermelha
   (luz vermelha)
=>
   (printout t "Pare" crlf))

(defrule luz-verde
   (luz verde)
=>
   (printout t "Siga" crlf))
```

Ao afirmar (luz vermelha), o robô imprime "Pare". Ao afirmar (luz verde), imprime "Siga".

DANDO UMA VOLTA

Mas há mais possibilidades: luzes com seta verde, sinais de “andar” e “não andar”, mãos vermelhas para pedestres. Assim, se o robô estiver andando, ele deve prestar atenção nesses sinais.

Podemos representar essa lógica com mais de um padrão por regra:

```
(defrule atravesse
   (status andando)
   (sinal-andar andar)
=>
   (printout t "Siga" crlf))
```

A regra só é ativada quando ambos os padrões estão presentes. Isso é um exemplo de **condição lógica AND** — todos os padrões devem ser verdadeiros para que a regra seja ativada.

ESTRATÉGIA DE EXECUÇÃO

CLIPS usa uma **estratégia de resolução de conflitos** para decidir qual regra ativar

quando múltiplas estão prontas. As estratégias disponíveis são: `depth`, `breadth`, `LEX`, `MEA`, `complexity`, `simplicity` e `random`.

A estratégia padrão é `depth`, na qual ativações mais recentes com mesma prioridade são executadas antes das mais antigas. Essa estratégia pode ser modificada, mas todos os exemplos neste guia assumem `depth`.

DICA: se você executar um sistema de outra pessoa, assegure-se de usar as mesmas configurações. O ideal é que as configurações sejam codificadas explicitamente no sistema.

USANDO DEFFACTS

Se estiver cansado de digitar os mesmos fatos, use `deffacts` para pré-carregá-los:

```
(deffacts andar "Fatos sobre andar"
   (status andando)
   (sinal-andar andar))
```

Esses fatos são inseridos com o comando (reset), que:

1. Remove fatos antigos e ativações.
2. Insere fatos definidos com `deffacts`.

Com isso, você pode limpar o ambiente sem apagar suas regras.

REMOVENDO DEFFACTS

Use `undeffacts` para remover um conjunto de fatos definidos:

(undeffacts andar)
(reset)

Isso evita que os fatos sejam reassertados. Para restaurar, você deve redefinir o `deffacts`.

MONITORANDO O SISTEMA

Você pode observar regras e ativações com:

(watch rules)
(watch activations)
(watch statistics)

Estatísticas incluem: tempo de execução, número de regras ativadas, média e máximo de fatos/ativações.

Use o comando:

(dribble-on "log.txt")

para registrar todas as entradas e saídas. Para parar:

(dribble-off)

O comando (run N) executa apenas N ativações, útil para depuração passo a passo.

PONTOS DE PARADA

Para depurar regras específicas, use breakpoints:

(set-break nome-da-regra)
(remove-break nome-da-regra)
(show-breaks)

CASAMENTO DE PADRÕES

Se uma regra não está ativando como esperado, use o comando (matches nome-da-regra) para verificar quais padrões casam com os fatos atuais.

Ele informará quais padrões são compatíveis, quais não são e se a regra está ativada.

Por exemplo:

(defrule ferias

```
   (trabalho feito)
   (dinheiro muito)
   (reservas feitas)
=>
   (printout t "Vamos viajar!" crlf))
```

```
(matches ferias)
```

...informará quais partes estão sendo satisfeitas pelos fatos atuais.

No próximo capítulo, vamos explorar o uso de variáveis para tornar os padrões ainda mais dinâmicos.

CAPÍTULO 4 – INTERESSES VARIÁVEIS

“Nada muda mais do que a mudança.”

Até aqui, vimos regras com padrões fixos. Agora vamos aprender como tornar as regras mais poderosas usando variáveis.

VAMOS VARIAR

Assim como em outras linguagens de programação, CLIPS possui variáveis para armazenar valores. A diferença é que, enquanto fatos são estáticos (não mudam após serem inseridos), variáveis são dinâmicas – seus valores podem mudar durante a execução da regra.

O nome de uma variável é sempre iniciado com um ponto de interrogação, seguido do nome:

?x

?sensor

?cor

?valvula

?sons-de-patos

Exemplos válidos de variáveis incluem: