به نام خدا

آموزش نرم افزار CLIPS

به زبان ساده – قسمت دوم

یادآوری

در قسمت اول شما یاد گرفتید چگونه حقایق را تعریف ***(assert)*** ، مشاهده ***(facts)*** و یا حذف ***(retract)*** کنید، همچنین نحوه تعریف قوانین ***(defrule)*** ، مشاهده قوانین ***(rules)*** و اجرای آنها ***(run)*** را با مثال‌های ساده فرا گرفتید.

نکته: سیستم‌های خبره متشکل از تنها یک قانون، زیاد کاربردی نیستند. سیستم های خبره کاربردی می‌توانند شامل صدها قانون باشند.

مثال: برنامه کاربردی شامل قوانین چندگانه که میخواهیم با توجه به رنگ چراغ راهنمایی دستور رفتن یا توقف را نمایش دهد.

```
CLIPS> (defrule red-light
(light red)
=>
(printout t "Stop" crif))

CLIPS> (defrule green-light
(light green)
=>
(printout t "Go" crlf))
```

در قانون اول اگرحقیقت چراغ قرمز (light red) وجود داشته باشد آنگاه پیام stop چاپ می‌شود و در قانون دوم اگر حقیقت چراغ سبز موجود باشد (light green) پیام Go چاپ خواهد شد. البته شما می توانید این قوانین را گسترش داده مثلا قانون چراغ زرد را هم اضافه کنید و یا این قوانین را برای عابر پیاده نیز تعریف کنید.

حقایق اولیه

گاهی اوقات نیاز داریم از یک تنظیم مشابه برای حقایق (facts)، بارها استفاده کنیم. به جای اینکه بارها آنها رو تایپ کنیم می توانیم از ساختار deffacts استفاده کنیم. این روش برای مشخص کردن حقایقی است که هر زمان بخواهیم بتوانیم آنها رو دوباره فراخوانی و بسازیم (reset).

- دستور ***deffacts*** : حقایق اولیه یا پیش فرض توسط این دستور تعریف می شوند

(Deffacts نام دسته حقایق (حقایق)**)**

مثال: تعریف یک دسته حقیقت اولیه به نام cars که شامل چندین حقیقت است.

CLIPS> **(deffacts Cars**
(car Pride) (car Reno) (car Samand))

شما در حال حاظر حقایق اولیه‌ای را تعریف کرده اید که در حافظه قرار دارند ولی هنوز آنها را ایجاد نکرده اید.

- دستور ***(reset)*** : هر زمان که نیاز بود می توانید حقایق اولیه را توسط این دستور بازیابی کنید.

CLIPS> **(reset)**

نکته: دقت کنید با اجرای این دستور کل سیستم دوباره راه اندازی می شود یعنی تمام حقایق تعریف شده پاک شده و فقط حقایق اولیه ایجاد می شوند.

- دستور ***undeffacts*** : توسط این دستور می توانید حقایقی اولیه‌ای که با دستور deffacts ایجاد شده اند را حذف کنید.

(undeffacts نام حقیقت اولیه **)**

مثال:

CLIPS> **(undeffacts Cars)**
CLIPS> **(reset)**

تطبیق الگو با حقایق

همان طور که به یاد دارید هر قاعده دارای الگو (شروط) می‌باشد. تا کنون، الگوهای مورد استفاده برای مطابقت قوانین با حقایق، بسیار ساده بوده است. در این حالت هر الگو با یک حقیقت خاص تطبیق داده می‌شود. اما با استفاده از 'نویسه جایگزین' می تواند قوانین را با چندین حقیقت تطبیق داده و عملیات مربوط به آنها بارها اجرا شوند.

مثال: فرض کنید حقایقی از حیوانات (گربه، سگ، اردک) تعریف کرده ایم

```
CLIPS> (deffacts animals
(animal  cat) (animal  dog) (animal duck))
CLIPS> (reset)
```

و حال می‌خواهیم یک قانون تعریف کنیم که با یافتن هر حیوان (animal) در لیست حقایق، پیام پیدا شد (animal found) را چاپ کند

```
CLIPS> (defrule animal
(animal ?)
=>
(printout t "animal found" crlf))
```

سپس قاعده تعریف شده را اجرا می‌کنیم

```
CLIPS> (run)
animal found
animal found
animal found
```

در مثال بالا الگو قاعده ما (animal ?) بوده که علامت ؟ به عنوان نویسه جایگزین می باشد (یعنی هر چیزی می تواند جای آن قرار گیرد). بنابراین با یافتن هر حقیقت مطابق با الگو، عملیات (در اینجا چاپ پیام) اجرا می‌شود. با اجرای قاعده سه حیوان در لیست حقایق یافت شده، بنابراین سه بار پیام animal found چاپ شده است.

متغیرها

در متغیر، از علامت ؟ (نویسه جایگزین) کنار نام متغیر استفاده میشود که توسط آن می‌توانید هر بار مقدار متغیر در قاعده را استخراج کنید.

نام متغیر?

مثال زیر را امتحان کنید: نام حیوانات موجود در لیست حقایق چاپ شود

```
CLIPS> (defrule list- animals
(animal  ?name)
=>
(printout t  ?name "found" crlf))
```

در صورت تطبیق الگو با حقایق، هر بار نام حیوان درون متغیر ?name قرار می‌گیرد و در خط آخر مقدار این متغیر چاپ می شود.

```
CLIPS> (run)
cat  found
dog  found
duck  found
```

بعد از اجرای این قاعده جواب بازگشتی CLIPS به شکل بالا خواهد بود.

نکته: همان طور که می‌دانید در هنگام حذف حقایق (retract) نیاز با دانستن شماره حقایق داریم. البته میتوانیم حقایق را بوسیله قوانین با اتصال متغیرها نیز حذف کنیم. مانند مثال زیر:

```
CLIPS> (defrule remove-animal
?fact <- (animal ?)
=>
(printout t "retracting" ?fact crlf)
(retract ?fact))
```

در قسمت الگو قاعده، متغیر ?fact با شماره ایندکس حقیقتی که با الگو (animal ?) تطبیق داشته باشد، مقداردهی می‌شود. علامت فلش به سمت چپ (<-) به همین منظور استفاده شده است.

بعد از اجرای این قاعده، حقایقی که با الگو این قاعده مطابق هستند حذف و یک پیام حذف نیز برایشان چاپ خواهد شد.

```
CLIPS> (run)
retracting<Fact-3>
<== f-3    (animal duck)
retracting<Fact-2>
<== f-2    (animal dog)
retracting<Fact-1>
<== f-1    (animal cat)
```

عملگرهای ریاضی و منطقی

- **عملگرهای منطقی**: همان طور که می‌دانید اگر در قوانین بیشتر از یک شرط (الگو) استفاده کنیم، بطور خودکار بین شروط عملگر منطقی AND قرار می‌گیرد به این معنی که تمام شروط باهم باید برقرار باشند تا قسمت عملیات اجرا شود، اما عملگرهای منطقی دیگری نظیر موارد زیر می‌تواند مورد استفاده قرار بگیرد:

not : اگر قبل از شری این عملگر را بگزارید، آن شرط را نقض می‌کنید (یعنی در صورت عدم وجود آن الگو)

Or : به معنای یا . اگر بین شروط قرار بگیرد در صورت درستی هر شرط، قسمت عملیات قاعده اجرا خواهد شد.

مثال:

```
(defrule take-umbrella
 (or (weather raining)
    (weather snowing))
 =>
 (assert (umbrella required)))
```

در این قاعده اگر 'هوا بارانی بود' یا 'هوا برفی بود' (یعنی حداقل یکی از این حقایق موجود باشد) آنگاه حقیقت چتر لازم است اضافه خواهد شد.

- **عملگرهای ریاضی**: در حالت عادی عملگرهای ریاضی بین دو آرگومان قرار می‌گیرد (حالت میاوندی یا infix)، به عنوان مثال 9+6 . اما در CLIPS این عملیات بصورت پیشوندی یا prefix می باشد یعنی ابتدا عملگر و بعد دو آرگومان قرار می‌گیرد. به عنوان مثال قبل باید به شکل (+ 9 6) نوشته شود.

مثال:

```
CLIPS> (+ 9 6)
15
CLIPS> (- 9 6)
3
CLIPS> (* 9 6)
54
CLIPS> (/ 9 6)
1.5
```

پایان قسمت دوم