الغاية من الجلسة: فهم العودية Recursion في البرولوغ.

مقدمة:

العودية (Recursion) هي مفهوم أساسي في البرمجة وعلوم الحاسوب، حيث تشير إلى فكرة أو تقنية تسمح لدالة أو تعبير باستدعاء ذاته أثناء تنفيذه. بشكل أساسي، يعني ذلك أن الدالة تقوم باستدعاء نفسها خلال تنفيذها.

هذا المفهوم يُعتبر أحد أسس التصميم والتنفيذ في البرمجة ويتيح للمطورين حلاً أنيقاً لحل مشاكل معينة، خاصة تلك التي تناسب بشكل جيد مع الهيكل العودي. في المفهوم الأساسي، يمكن تفسير العودية على أنها "تعليمة أو تقنية تستخدم دالة لتنفيذ نفسها بشكل متكرر حتى تحقق شرط معين للتوقف".

منفعة العودية تظهر بشكل واضح عندما تكون المشكلة قابلة للتقسيم إلى مشاكل أصغر من نفس النوع. في هذه الحالة، يمكن للدالة الاستدعاء نفسها لحل المشاكل الفرعية، مما يُسهم في تنظيم الشيفرة وجعلها أكثر فهما وصيانة.

شروط العودية:

تتطلب العودية (Recursion) وجود شروط معينة لضمان توقف الاستدعاء المتكرر وتجنب حدوث حلقة لانهائية. هذه الشروط الأساسية تشمل:

1. حالة الأساس (Base Case) وفي البرولوغ تسمى (Base Clause): يجب أن يكون هناك حالة خاصة تؤدي إلى توقف الاستدعاء العودي. عندما تحقق هذه الحالة، يتوقف الاستدعاء وتعود قيمة محددة.
2. عبارة العودية: وتعني أن تستدعي الطريقة نفسها أي في البرولوغ في قاعدة المعرفة، تتحقق العودية عندما يكون رأس القاعدة موجود في جسمها.
3. تغيير الحالة (Progress Toward Base Case): في كل مرة يتم فيها استدعاء الدالة العودية، يجب أن يحدث تغيير في الحالة نحو الوصول إلى الحالة الأساس.
4. عدم وجود حلقة لانهائية (Avoid Infinite Loop): يجب تصميم الدالة بحيث تتأكد من أنها ستتجه نحو حالة الأساس ولا تؤدي إلى حلقة لانهائية من الاستدعاء.

عند تحقيق هذه الشروط، يمكن للعودية أن تكون آلية قوية وفعالة لحل مجموعة متنوعة من المشاكل.

مثال:

لدينا قاعدة المعرفة الآتية:

child(martha,charlotte).

child(charlotte,caroline).

child(caroline,laura).

child(laura,rose).

descend(X,Y) :- child(X,Y).

descend(X,Y) :- child(X,Z),descend(Z,Y).

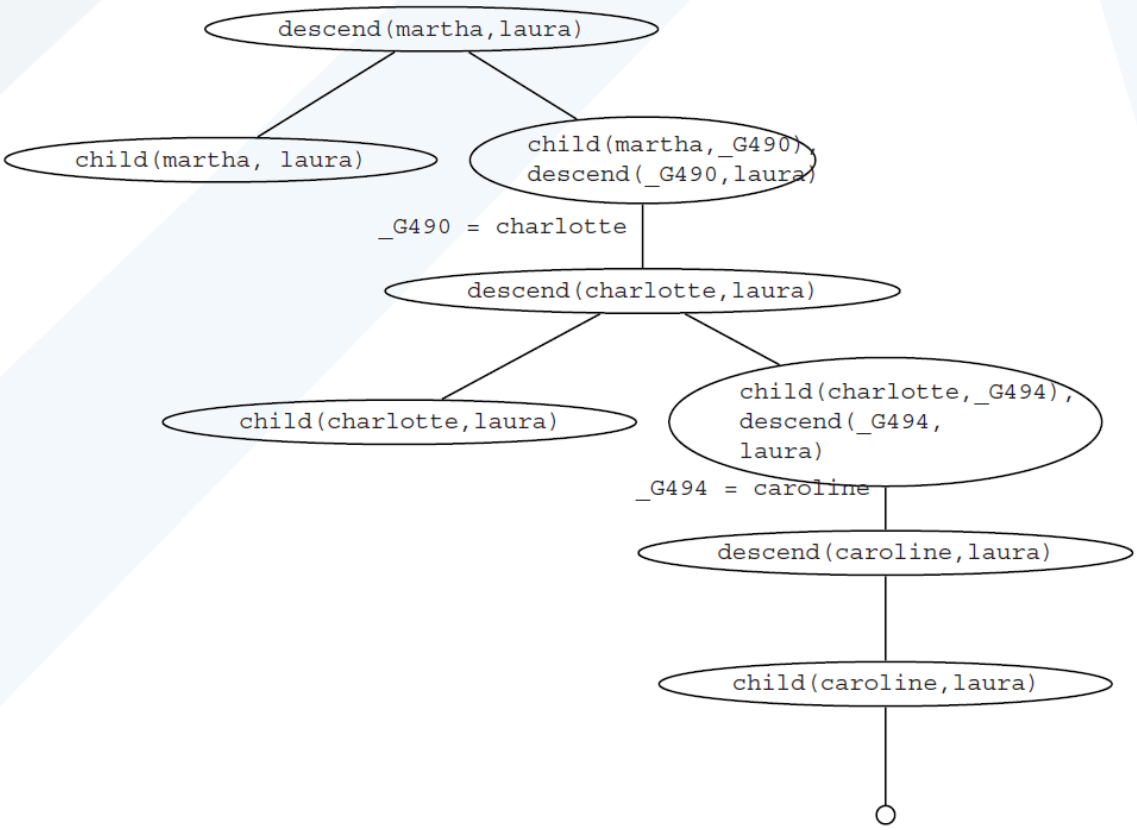
والمقصود بالـ child أن الوسيط الثاني هو ابن للوسيط الأول أي مثلاً في أول حقيقة فإن charlotte هي بنت martha.

أما descend فتعني السلف أي descend(X,Y) تعني أن Y هو من نسل أو منحدر من X (أي أن Y هو ابن أو حفيد أو ابن الحفيد أو .... بالنسبة لـ X).

لو طرحنا الاستعلام الآتي في البرولوغ:

?- descend(martha,laura).

فإن البرولوغ من أجل أن يجاوبك على السؤال السابق فإنه سوف يرسم ضمنياً شجرة البحث هذه:



طبعاً عندما يصل إلى الدائرة الفارغة سوف يعطي true ولكن لن يتوقف الرسم هنا، أقصد أن للرسمة تتمة ويمكنك أن تكمل رسمها بنفسك، حيث أن هناك خيار آخر لـ descend(caroline, laura) وهذا الخيار لم يجربه البرولوغ في هذا المثال (وهو القاعدة الثانية طبعاً).

مثال:

numeral(0).

numeral(succ(X)) :- numeral(X).

وطرحنا الاستعلام الآتي: ?- numeral(X).

كيف سيجيب البرولوغ عن السؤال السابق؟

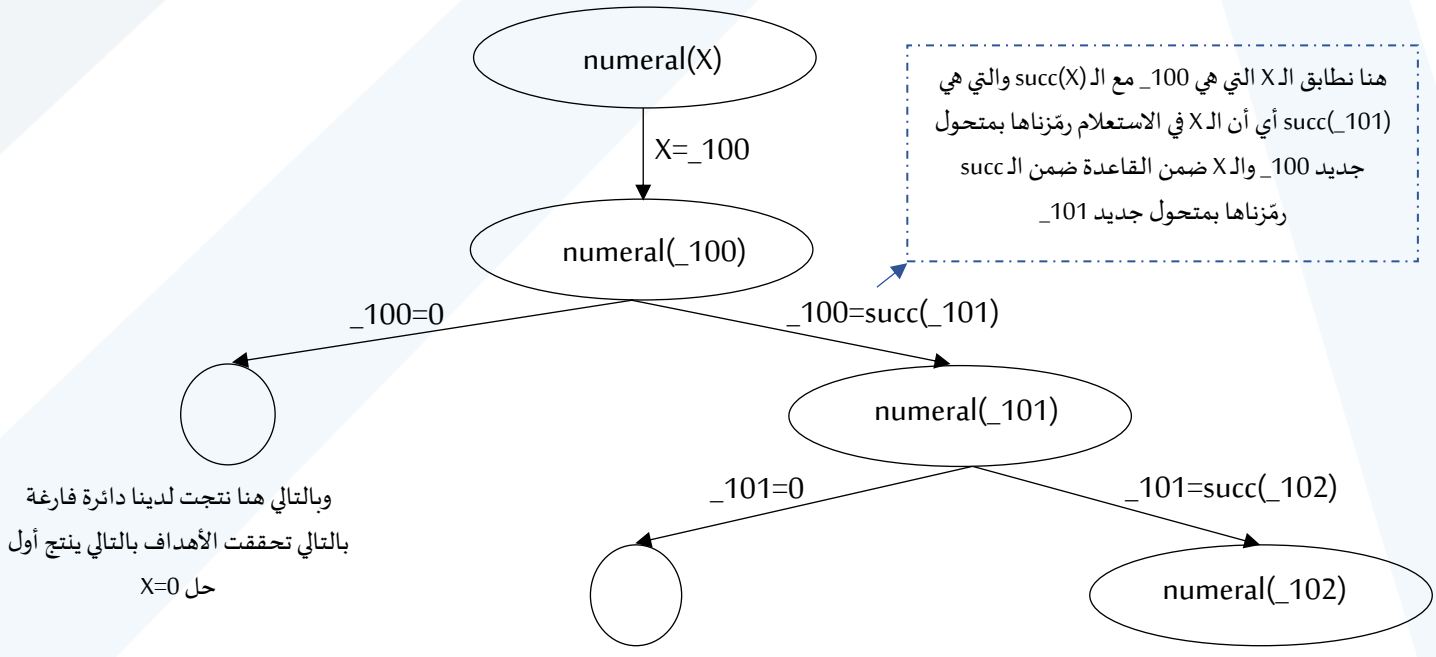
إن قاعدة المعرفة السابقة تحوي حقيقة تقول أن الـ 0 هو **عدد numeral**، وتحوي قاعدة تقول أن العدد X+1 (أي succ(X)) هو **عدد** عندما يكون X هو **عدد**.

إن المُعلَن succ(X) يعني العدد الذي يزيد X بواحد، وطبعاً هي ليست معرفة في قاعدة المعرفة وهذا مقبول.

الآن، ما هو جواب البرولوغ عن الاستعلام السابق؟

```
X = 0 ;
X = succ(0) ;
X = succ(succ(0)) ;
X = succ(succ(succ(0))) ;
X = succ(succ(succ(succ(0)))) ;
X = succ(succ(succ(succ(succ(0))))) ;
X = succ(succ(succ(succ(succ(succ(0)))))) ;
X = succ(succ(succ(succ(succ(succ(succ(0))))))) ;
X = succ(succ(succ(succ(succ(succ(succ(succ(0)))))))) ;
X = succ(succ(succ(succ(succ(succ(succ(succ(succ(0))))))))) ;
```

لنحاول أن نرسم الـ proof search tree:



وهكذا نكمل بنفس الطريقة.