HOE TEKEN IK EEN OMGEVINGSMODEL

MATTIAS DE WAEL, WARD MUYLAERT

1. Inleiding

Om de allereenvoudigste Scheme expressies te begrijpen volstaat het substitutiemodel. Het substitutiemodel veronderstelt het bestaan van een "boekje" dat (namen van) variabelen bindt met een waarde. Wanneer een variabele wordt gebruikt, wordt die simpelweg vervangen, of gesubstitueerd, door de waarde uit de boekhouding. Wanneer de Scheme expressies complexer worden, en vooral als er procedures en hogere orde procedures in het spel zijn, schiet het substitutiemodel tekort. Om de hiaten van het substitutiemodel op te vangen maken we gebruik van het omgevingsmodel, wat een uitbereiding is van het substitutiemodel in die zin dat het "boekje" met variabelen bindingen nu ook "deelboekjes" met bindingen kan bevatten. We noemen deze hiërarchische "boekjes" ook wel omgevingen vandaar ook de naam omgevingsmodellen.

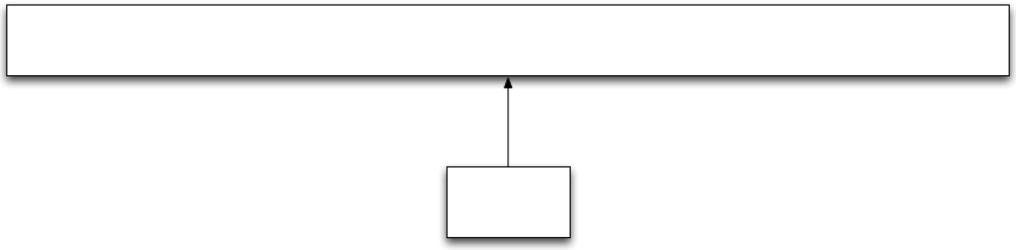
2. Onderdelen

Om een omgevingsmodel te tekenen heb je enkele *bouwstenen* nodig waaruit het volledige schema is opgebouwd.

2.1. **Omgeving.** Hét basis onderdeel van een omgevingsmodel is de omgeving. Een omgeving wordt voorgesteld door een eenvoudige rechthoek. In deze rechthoek zullen alle bindingen van variabelen aan hun waarde worden weergegeven.

2.1.1. *De Globale Omgeving.* Het enige item dat in alle omgevingsmodellen voorkomt is de globale omgeving. De globale omgeving is typisch leeg aan het begin van de REPL, of bevat de primitieve en ingebouwde functies. De globale omgeving is een omgeving en wordt dus door een rechthoek voorgesteld. Gewoonlijk wordt de rechthoek over volledige breedte van de tekening getekend.

2.1.2. *De Lokale Omgeving.* Bij elke procedure oproep wordt een nieuwe lokale omgeving aangemaakt, zo een lokale omgeving is een omgeving en wordt dus ook met een rechthoek voorgesteld. De rechthoeken worden evenwel een stuk kleiner getekend dan de rechthoeken voor de globale omgeving. Om aan te geven onder welke omliggende omgeving een lokale omgeving ligt, wordt een pijl getekend van de lokale naar de omliggende omgeving.



FIGUUR 1. De globale omgeving met een lokale omgeving die zich erin bevindt.

2.2. **De procedure.** `(lambda (n) (* 2 n))` De lambda expressie evalueert in Scheme naar een procedure. Om een procedure(-object) volledig te definiëren, moet je drie zaken kennen:

- De lijst van formele parameters;
- De body van de procedure; en
- De omgeving waarin de procedure gedefinieerd is.

Het procedure object zelf stellen we voor door twee horizontaal aan elkaar geplakte cirkels, die je typisch vlak onder de omgeving tekent waar de procedure wordt gedefinieerd. Vervolgens verbind je de rechter van de twee cirkels met die omgeving. Vanuit de linker cirkel laat je een pijl vertrekken naar een plaats op je schema, liefst zo dicht mogelijk bij de cirkels, waar je plaats hebt om de parameter lijst en de body neer de schrijven, die je respectievelijk laat voorafgaan door "P :" en "B :".

Veronderstellen we dat de lambda expressie in de globale omgeving gedefinieerd is, dan kunnen we de lambda-expressie voorstellen zoals in Figuur 2.

FIGUUR 2. Globale omgeving met een procedure-object voor de lambda-expressie `(lambda (n) (* 2 n))` .

Omdat de parameter lijst en body veelal vlak naast de twee cirkels worden getekend, wordt de pijl vanuit de linker cirkel vaak achterwege gelaten. De pijl uit de rechter cirkel is een noodzakelijk onderdeel van het schema omdat deze de omgeving aanduidt waar de procedure gedefinieerd is.

2.3. **Cons-cellen.** Hoewel het niet vaak voorkomt in omgevingsmodellen kan je cons-cellen voorstellen door hun box-and-pointer representatie.

2.4. **De Variabelen Binding.** Zonder variabelen bindingen zegt een omgeving natuurlijk niet veel. Een variabelen binding is de binding van de naam van een variabele en haar waarde. We stellen ze voor in het omgevingsdiagram door in de omgeving waar de binding bestaat de variabelen naam te schrijven gevolgd door een dubbelpunt. Na het dubbelpunt volgt de waarde van de variabele.

2.4.1. *Directe Waarde.* Als de waarde van de variabele een directe waarde is zoals een symbool, een getal, een string, ..., dan kunnen we die waarde direct na de dubbele punt schrijven. `(define aantal 5)` wordt dus eenvoudigweg: "aantal : 5".

2.4.2. *Procedure-objecten als waarde.* Om aan te duiden dat een variabele gebonden is aan een procedure object, laat je een pijl vertrekken van aan het dubbelpunt naar de twee cirkels die het procedure object voorstellen. Om te begrijpen hoe we `(define (verdubbel n) (* 2 n))` weergeven, haal je best even de syntactische suiker weg van deze expressie. `(define verdubbel (lambda (n) (* 2 n)))` Dit toont iets duidelijker wat er juist gebeurt: er wordt een procedure aangemaakt, die dan aan de variabele "verdubbel" wordt gebonden. We zullen dus in de omgeving "verdubbel : $\rightarrow$" zien staan.

2.4.3. *Herbinding.* Scheme is geen zuivere functionele taal, er kunnen ook destructieve operaties worden uitgevoerd, zoals bijvoorbeeld `(set! aantal 6)` . In het omgevingsmodel zullen we dit voorstellen door de oude waarde te doorstrepen en de nieuwe er naast te schrijven. "aantal : 5" wordt dus "aantal : ~~5~~ 6". Let op met herbindingen. Een omgevingsmodel geeft de situatie weer van het geheugen op een specifiek moment. Door een oude waarde te vervangen door een nieuwe, kan het soms verwarrend zijn, als je je schema herinterpreteert, om te achterhalen waar een bepaalde waarde vandaan komt.

Als voorbeeld de volgende code en bijhorend omgevingsmodel.

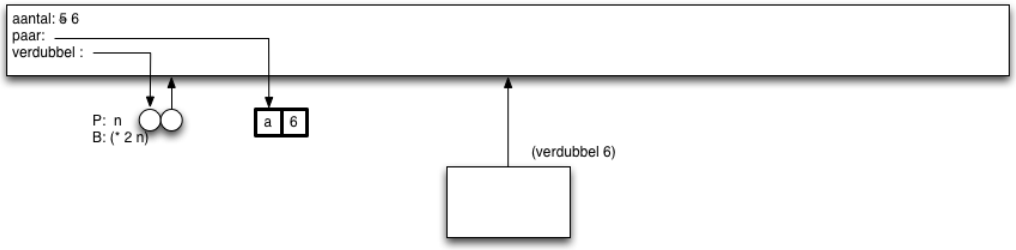
```
(define aantal 5)
(define paar (cons 'a 6))
(define (verdubbel n) (* 2 n))
(set!  aantal 6)
```

FIGUUR 3. Omgevingsmodel van Scheme code met een cons-cell, een procedure-object en een herbinding.

3. De procedure oproep

Nu we alle bouwstenen kennen voor het tekenen van een omgevingsmodel, kunnen we verder gaan en het omgevingsmodel gebruiken waarvoor het eigenlijk dient, namelijk uitleggen wat er gebeurt bij het uitvoeren van een Scheme programma met respect tot het geheugen en de scope van variabelen. De grootste veranderingen aan het schema gebeuren bij het uitvoeren van een procedure. Veronderstel een globale omgeving waarin alle definities zijn gebeurd zoals in de vorige sectie. Veronderstel verder dat dan de volgende expressie wordt uitgevoerd: `(verdubbel 6)` .

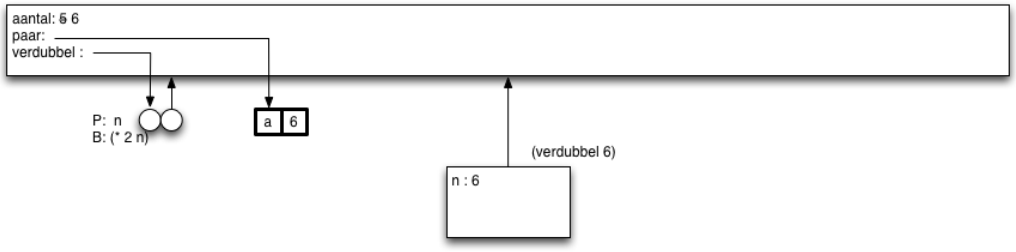
3.1. **Een nieuwe lokale omgeving.** Het eerste wat er gebeurt als een procedure wordt uitgevoerd is het aanmaken van een nieuwe lokale omgeving die onder de omgeving wordt gehangen waar de op te roepen procedure is gedefinieerd. Vervolgens worden alle argumenten geëvalueerd, die op zich weer complexe expressies kunnen zijn. Voor dit voorbeeld houden we het simpel en roepen we `verdubbel` op met 6 als argument.



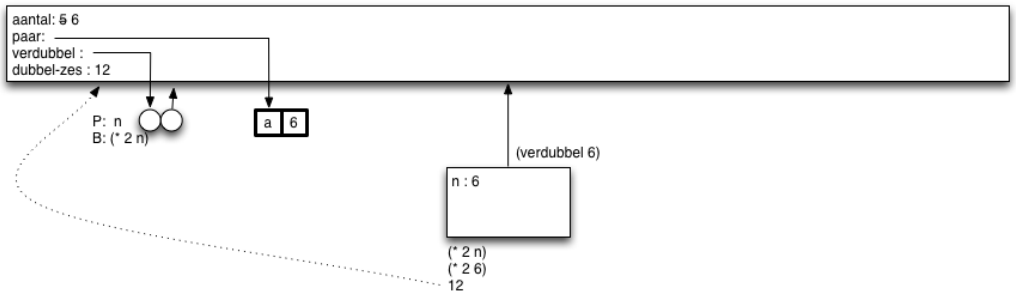
Figuur 4. Een nieuwe lokale omgeving wordt toegevoegd aan het model van Figuur 3.

3.2. **Bind de actuele parameters aan de formele parameters.** Als de lokale omgeving is aangemaakt en alle argument zijn geëvalueerd tot directe waarden of waarden die op de schema getekend zijn, nemen we de formele parameter lijst over uit de opgeroepen procedure en voegen elke parameter toe aan de lokale omgeving zoals we zouden doen voor variabelen. In dit voorbeeld met slecht 1 parameter, schrijven we dus “n :”. Vervolgens binden we de actuele waarde aan de formele parameter en krijgen we dus “n : 6”.

3.3. **De body uitvoeren.** Als laatste stap neem je de body van de uit te voeren procedure over, door ze links onder de lokale omgeving te schrijven. Werk de expressie zover mogelijk uit tot je een waarde bekomt. Deze waarde is de return-waarde van de procedure oproep, maar er bestaat geen manier om dat duidelijk weer te geven in het omgevingsmodel. Een mogelijk hulpmiddel kan een stippellijn-pijl zijn. Zodat `(define dubbel-zes (verdubbel 6))` het schema in Figuur 6 oplevert.



FIGUUR 5. We binden de actuele parameter 6 aan de formele parameter n.



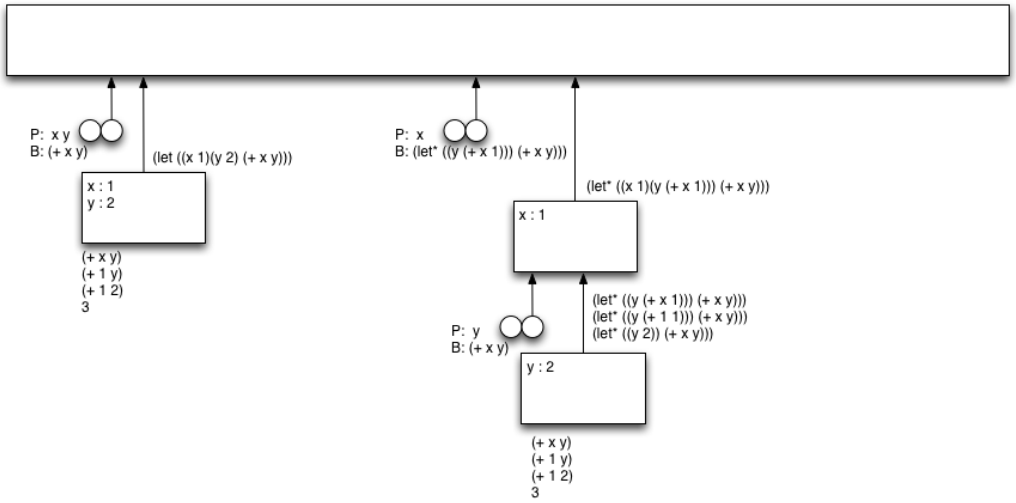
FIGUUR 6. We voeren de body van de procedure uit en duiden de return waarde aan.

4. De let-expressie

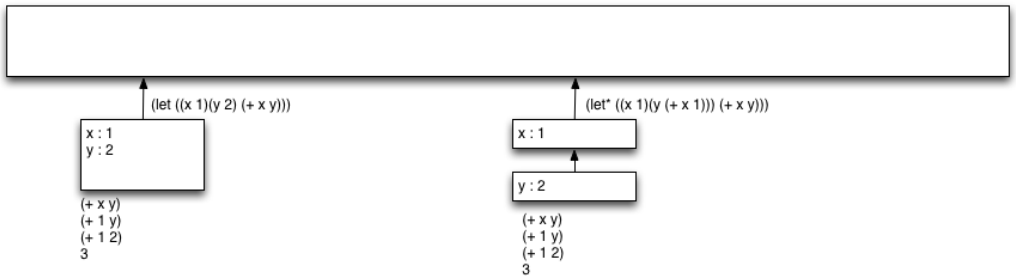
4.1. **De let expressie.** De let-expressie is syntactische suiker voor een direct uitgevoerde lambda. De expressie `(let ((x 1)(y 2)) (+ x y))` kan dus ook geschreven worden als `((lambda (x y) (+ x y)) 1 2)` . Deze laatste expressie geeft aanleiding tot het schema links in Figuur 7.

4.2. **De let* expressie.** Ook de expressie `(let* ((x 1)(y (+ x 1))) (+ x y)))` is syntactische suiker, namelijk voor de geneste lambda's: `((lambda (x) ((lambda (y) (+ x y)) (+ x 1))) 1)` . Dit staat afgebeeld rechts in Figuur 7.

4.3. **Versimpelde tekening.** Omdat de let-expressie een fundamenteel en vaak voorkomend onderdeel is in Scheme programma's wordt de expliciete anonieme procedure vaak achterwege gelaten in omgevingsmodellen, en wordt er direct een lokale omgeving getekend met de bindingen uit de let. Dit wordt weergegeven in Figuur 8.



FIGUUR 7. Omgevingsmodel voor de let (links) en let* (rechts) met expliciete anonieme procedures.



FIGUUR 8. Versimpeld omgevingsmodel voor de let (links) en let* (rechts) met impliciete anonieme procedures.