6 Moleculaire Dynamica

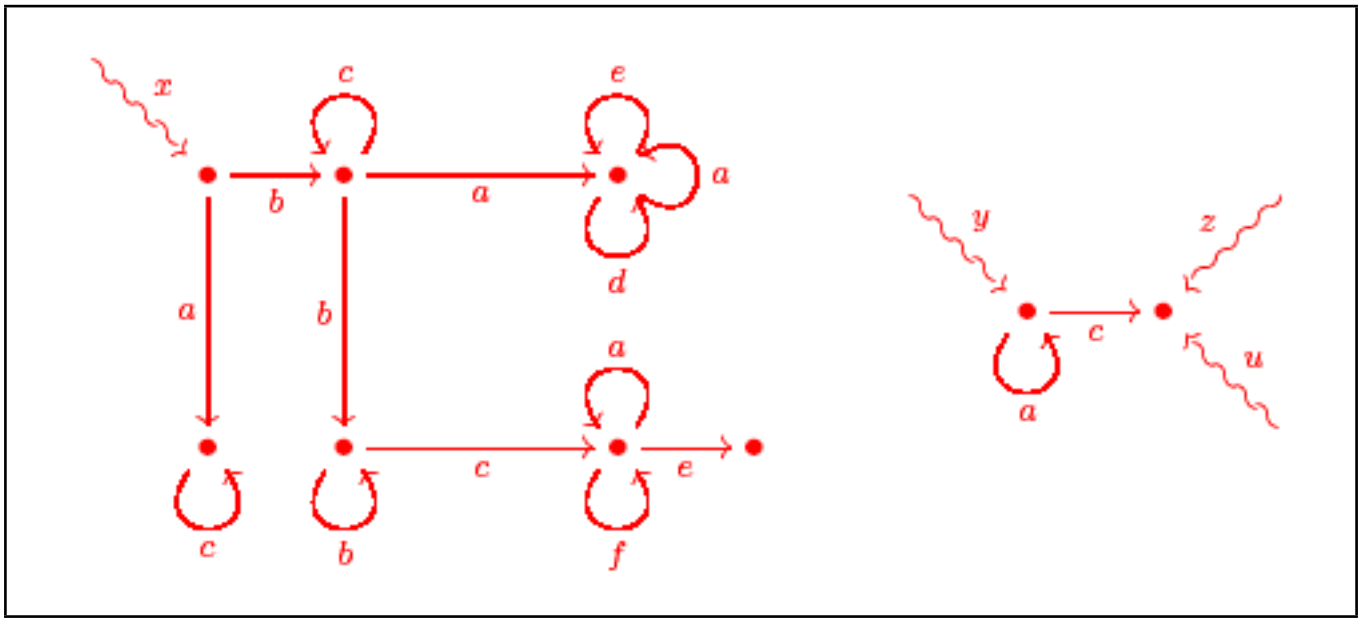
- Atomen, moleculen en vloeistoffen
- Getallen als moleculen
- Geheugensanering
- Waarden en schakelingen

6.1 Atomen, moleculen en vloeistoffen

- Toestand (van een computerprogramma): vloeistof bestaande uit een aantal moleculen, d.w.z.
  - ⋆ *atomen*,
  - ⋆ *velden*.
- Toestandsverandering: het aangaan van nieuwe of het verbreken van bestaande bindingen, d.w.z.
  - ⋆ toevoegen of verwijderen van velden,
  - ⋆ veranderen van het doel van velden.
- Observatie van toestanden en veranderingen: *focus*.

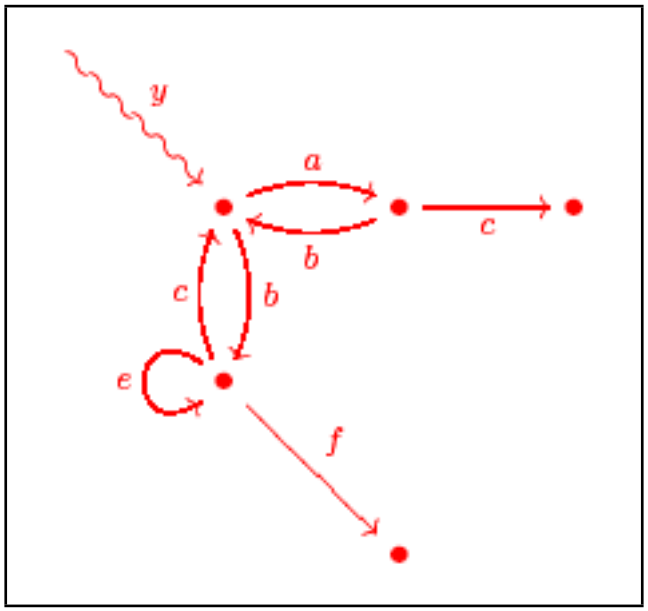
6.1 Atomen, moleculen en vloeistoffen

Eenvoudige diagrammen: vloeistof bestaande uit 2 moleculen



6.1 Atomen, moleculen en vloeistoffen

Molecule met dubbele bindingen



6.2 Instructies en acties

MPP (molecular programming primitives): 4 mutaties, 2 toekenningen en 2 tests.

Mutaties.

$Atoomcreatie\ \ x = new$

$Veldintroductie\ \ x.{+}f$

$Veldverwijdering\ \ x.{-}f$

$Veldmutatie\ \ x.f = y$

6.2 Instructies en acties

Toekenningen.

$Veldselectie\ x = y.f$

$Toekenning\ x = y$

Tests.

$Atoomidentiteitstest\ x == y$

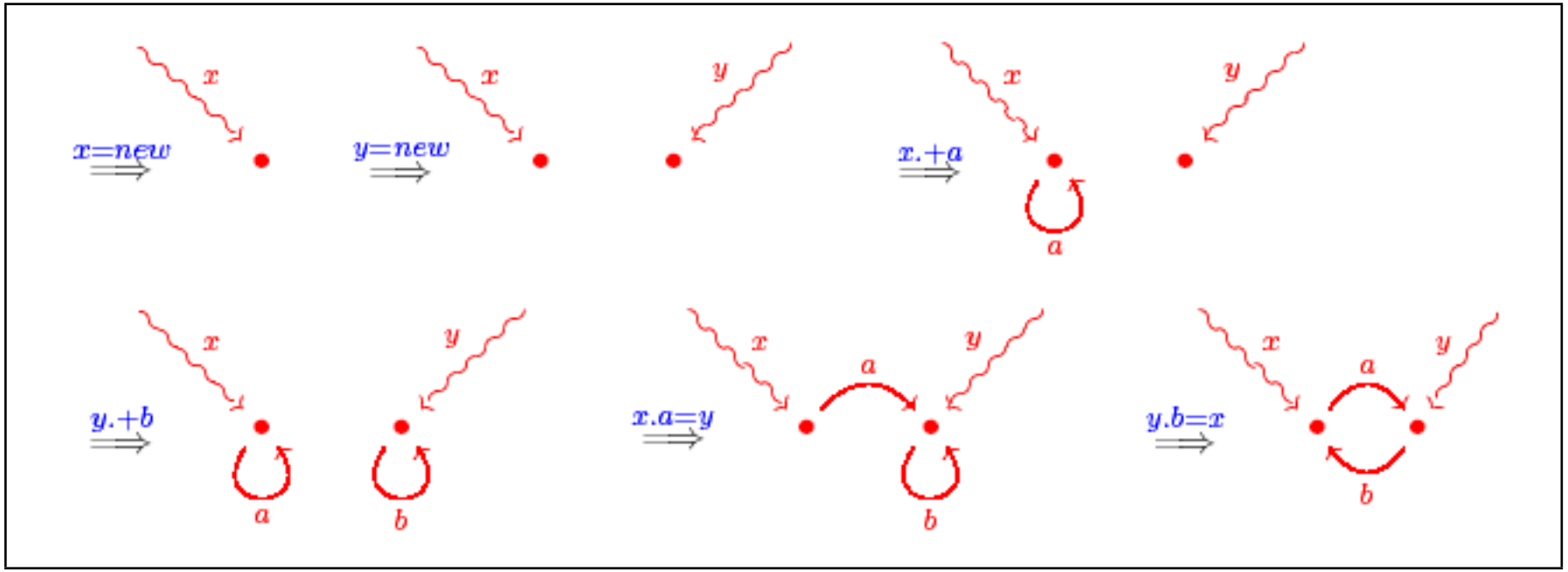
$Veldtest\ x/f$

Merk op: MPP heeft als impliciete parameters de collecties van focusnamen en veldnamen. Beide zijn oneindig.

6.2 Instructies en acties

Aaneenschakeling van instructies:

$$X \quad = \quad x = new; y = new; x.+a; y.+b; x.a = y; y.b = x.$$

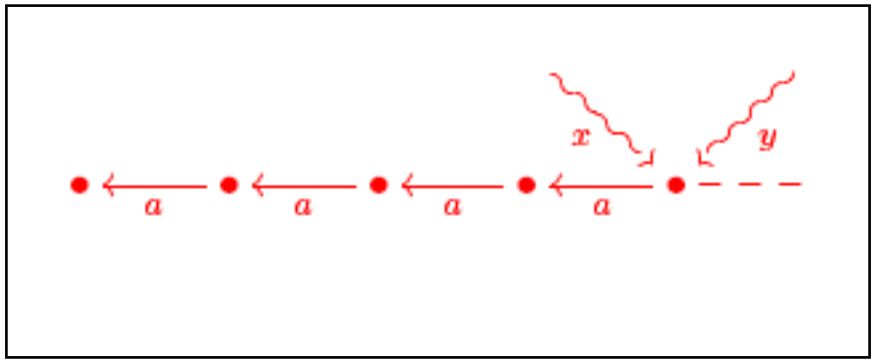


6.2 Instructies en acties

Met $A = \mathrm{MPP}$ worden de programmeertalen PGLA.mpp, PGLB.mpp, ... verkregen.

Voorbeeld in PGLDg.mpp:

$$x = new; \pounds 0; y = new; y.{+}a; y.a = x; x = y; \#\#\pounds 0$$



6.3 Rekenen zonder getallen

Model 1 (in PGLEc.mpp): $setZero$ creëert de 0, $pred$ berekent de voorganger en $succ$ de opvolger.

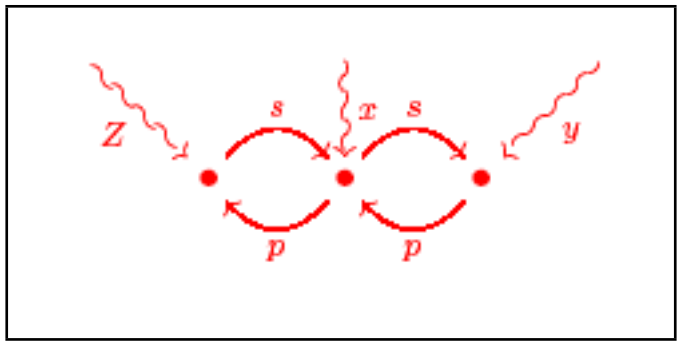
$$
\begin{aligned}
setZero &= Z = new; y = Z \\
pred &= -x/p\{; y = x; \}\{; y = x.p; \} \\
succ &= -x/s\{; x.{+}s; y = new; y.{+}p; y.p = x; x.s = y; \\
&\quad \}\{; y = x.s; \}
\end{aligned}
$$

Afspraak: de argumenten van unaire (binaire) functies hebben de focusnaam x ($x1$ en $x2$), het resultaat heet y.

6.3 Rekenen zonder getallen

Voorbeeld (x representeert de 1 en y de 2):

$$setZero; x = y; succ; x = y; succ$$



6.3 Rekenen zonder getallen

Optelling in model 1:

$$add \quad = \quad \pounds 0; +x2 == Z\{; y = x1; \}\{; x = x1; succ; x1 = y;$$

$$x = x2; pred; x2 = y; \#\#\pounds 0; \}$$

Merk op: de implementatie van add correspondeert met de recursieve definitie

$$x1 + x2 = \begin{cases} x1 & \text{als } x2 = 0, \\ (x1 + 1) + (x2 - 1) & \text{anders.} \end{cases}$$

6.3 Rekenen zonder getallen

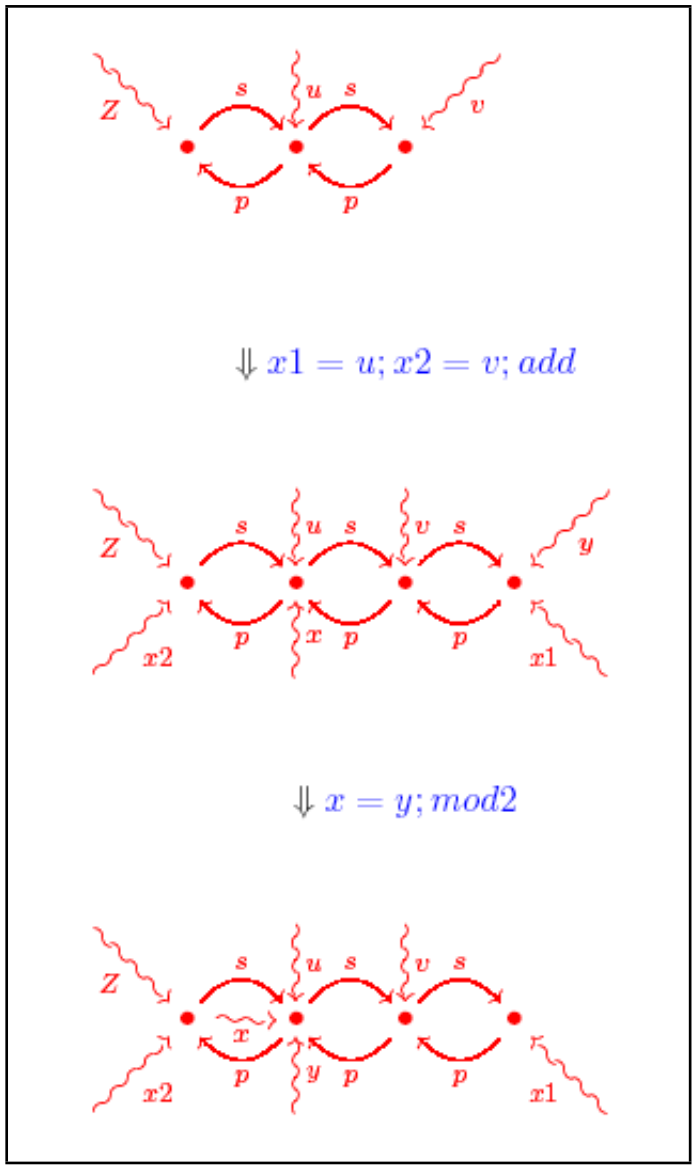
"modulus 2" in model 1:

$$mod2 \quad = \quad y = x; \pounds 1; -y == Z\{; x = y; pred;$$

$$-y == Z\{; x = y; pred; \#\#\pounds 1; \}\{; y = x; \}; \}\{; \}$$

Merk op: de implementatie van $mod2$ correspondeert met de recursieve definitie

$$x \mod 2 = \begin{cases} 0 & \text{als } x = 0, \\ (x-2) \mod 2 & \text{als } x - 1 > 0, \\ x & \text{anders.} \end{cases}$$


Z
s
u
s
v
p
p
⇓ x1 = u; x2 = v; add
Z
s
u
s
v
s
y
x2
p
x
p
p
x1
⇓ x = y; mod2
Z
s
u
s
v
s
x
x2
p
y
p
p
x1

(1+2) mod 2

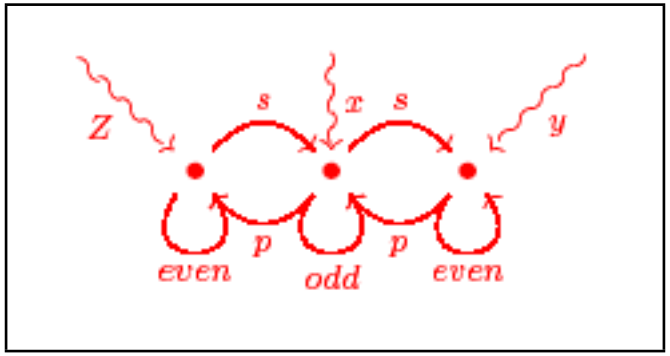
6.3 Rekenen zonder getallen

Model 2 (in PGLEc.mpp): uitbreiding van model 1 met $even$- en odd-velden

$$
\begin{aligned}
setZero &= Z = new; Z.{+}even; y = Z \\
pred &= -x/p\{;y = x;\}\{;y = x.p;\} \\
succ &= -x/s\{;x.{+}s; y = new; y.{+}p; y.p = x; x.s = y; \\
&\qquad +x/even\{;y.{+}odd;\}\{;y.{+}even;\};\}\{;y = x.s;\}
\end{aligned}
$$

6.3 Rekenen zonder getallen

De getallen 0, 1 en 2 in model 2:



$mod2$ is in model 2 eenvoudig:

$$+x/even\{;y = Z;\}\{;y = Z.s;\}.$$

6.4 Geheugensanering

Garbage- en non-garbage atomen:

Basis: Ieder door een focus geselecteerd atoom is non-garbage.

Inductie: Als het atoom in de oorsprong van een binding non-garbage is, dan is ook het doel non-garbage.

Afsluiting: Een atoom is alleen non-garbage als het uit de basis verkregen kan worden door een eindig aantal malen de inductiestap toe te passen.

Garbage-atomen zijn atomen die niet non-garbage zijn.

6.4 Geheugensanering

We veronderstellen dat

- atomen met een *referentie-teller* worden opgeslagen voor het bijhouden van het aantal inkomende foci en bindingen,

- een uniek atoom $null$ bestaat dat overbodige foci selecteert.

6.4 Geheugensanering

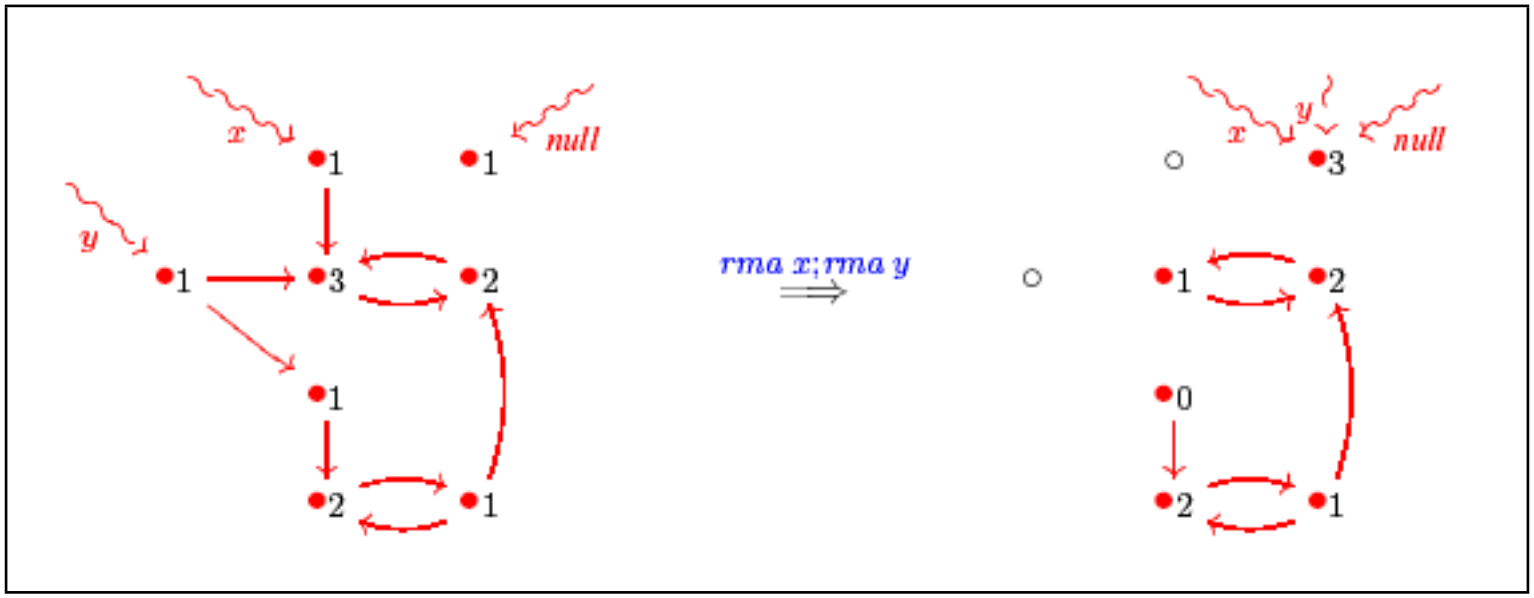
Drie instructies voor geheugensanering

Atoomverwijdering : (*remove atom*) $rma\ x$.

Beperkte garbage collection : (*restricted garbage collection*) rgc.

Volledige garbage collection : (*full garbage collection*) fgc.

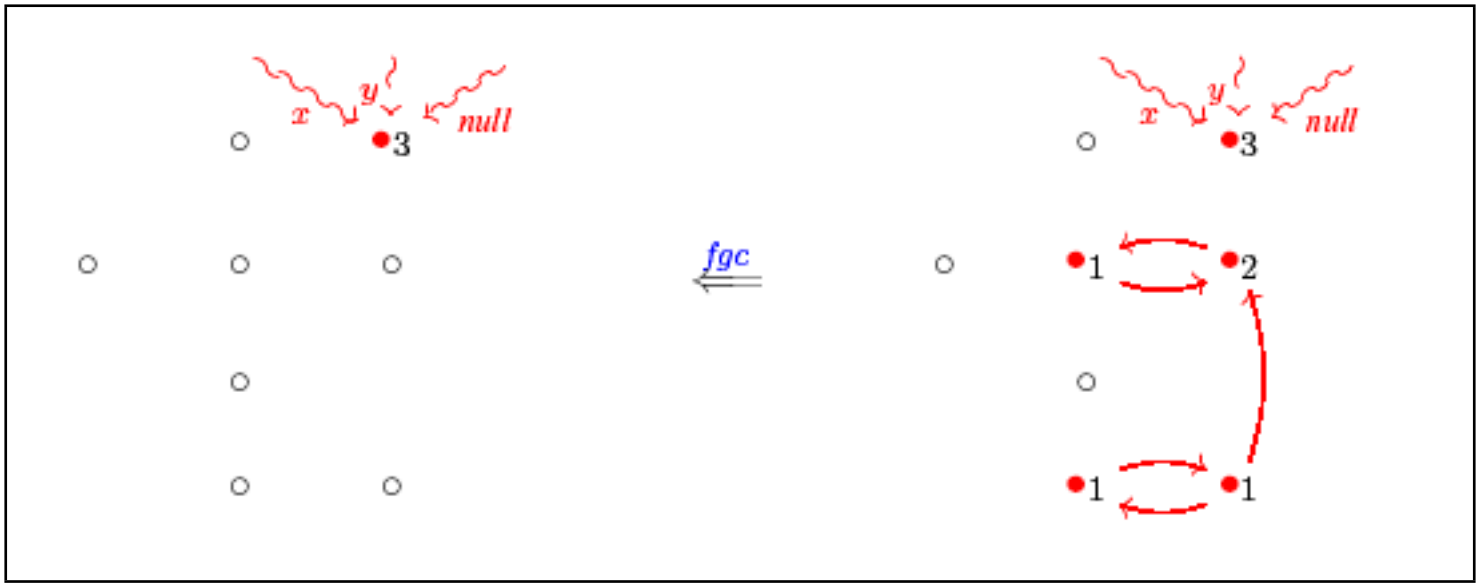
6.4 Geheugensanering



$\Downarrow rgc$

6.4 Geheugensanering

$\Downarrow rgc$



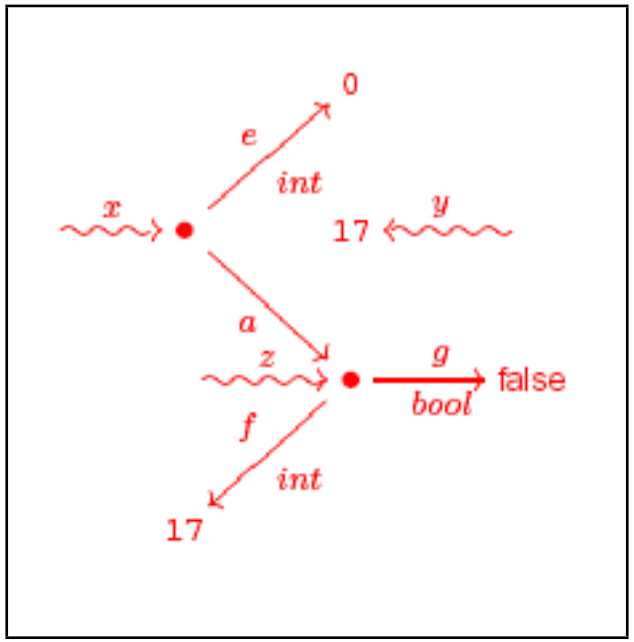
6.5 Waarden

Waarden:

- vast en eindig aantal *basistypen* : $bool$, int, ...,
- *literalen* (benaming van waarden): true, false, ..., 0, 1, ...,
- waarden worden beschouwd als terminale objecten, d.w.z. objecten zonder uitgaande verbindingen.

6.5 Waarden

Een molecule met waardevelden:



6.5 Waarden

MPPV is de uitbreiding van MPP met 4 mutaties

$Waardeveldintroductie$: $x.+f : t$.

$Waardeveldverwijdering$: $x.-f : t$.

$Waardeveldmutatie$: $x.f = y$.

$Waardeveldmutatie\ met\ constante$: $x.f = u$.

6.5 Waarden

en 3 toekenningen en 3 tests

$Waardeveldselectie$: $x = y.f$.

$Waardetoekenning$: $x = y$.

$Waardetoekenning\ met\ constante$: $x = u$.

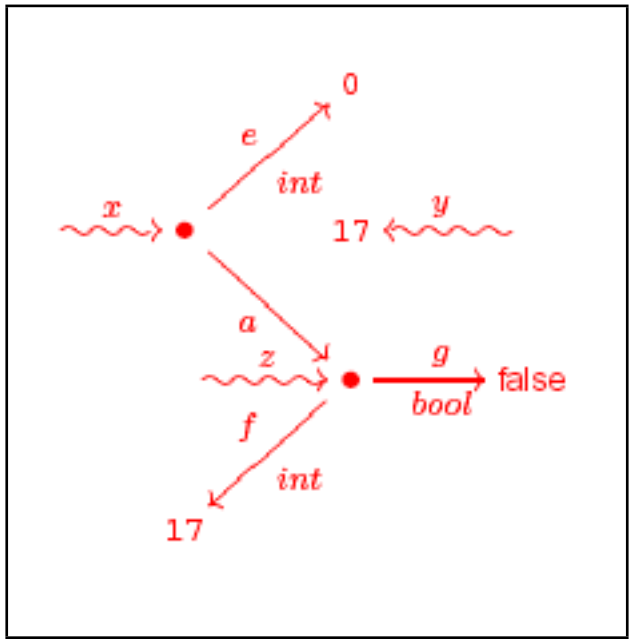
$Waarde\text{-}identiteitstest$: $x == y$.

$Waarde\text{-}identiteitstest\ met\ constante$: $x == u$.

$Waardeveldtest$: $x/f : t$.

6.5 Waarden

Voorbeeld:



$$x = new; z = new; y = 17; x.{+}e : int; x.{+}a; z.{+}g : bool;$$
$$x.e = 0; x.a = z; z.g = \mathsf{false}; z.{+}f : int; z.f = y$$

6.6 Rekenen met waarden

Waarheidstafels voor negatie en conjunctie:

i	negatie
1	0
0	1

$i1$	$i2$	conjunctie
1	1	1
1	0	0
0	1	0
0	0	0

6.6 Rekenen met waarden

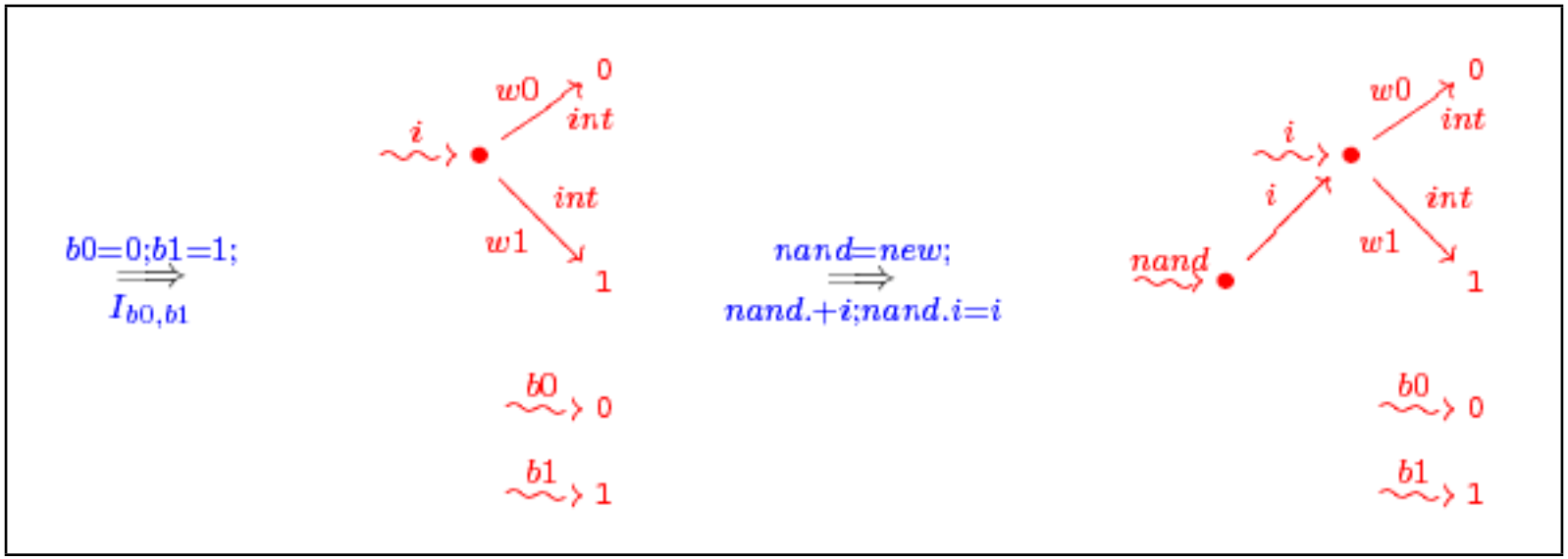
Basisprogramma's NEG, AND en $I_{b0,\ldots,bn}$:

$$
\begin{aligned}
NEG \quad &= \quad neg = new; neg.{+}i; neg.i = i; o = new; o.{+}w0 : int; \\
&\quad\; i0 = i.w0; {+}i0 == 1\{; o.w0 = 0; \}\{; o.w0 = 1; \}; \\
&\quad\; neg.{+}o; neg.o = o
\end{aligned}
$$

$$
\begin{aligned}
AND \quad &= \quad and = new; and.{+}i; and.i = i; o = new; o.{+}w0 : int; \\
&\quad\; i0 = i.w0; i1 = i.w1; \\
&\quad\; {+}i0 == 1\{; {+}i1 == 1\{; o.w0 = 1; \}\{; o.w0 = 0; \}; \}\{; \}; \\
&\quad\; and.{+}o; and.o = o
\end{aligned}
$$

$$
I_{b0,\ldots,bn} \quad = \quad i = new; i.{+}w0 : int; i.w0 = b0; \ldots; i.{+}wn : int; i.wn = bn
$$

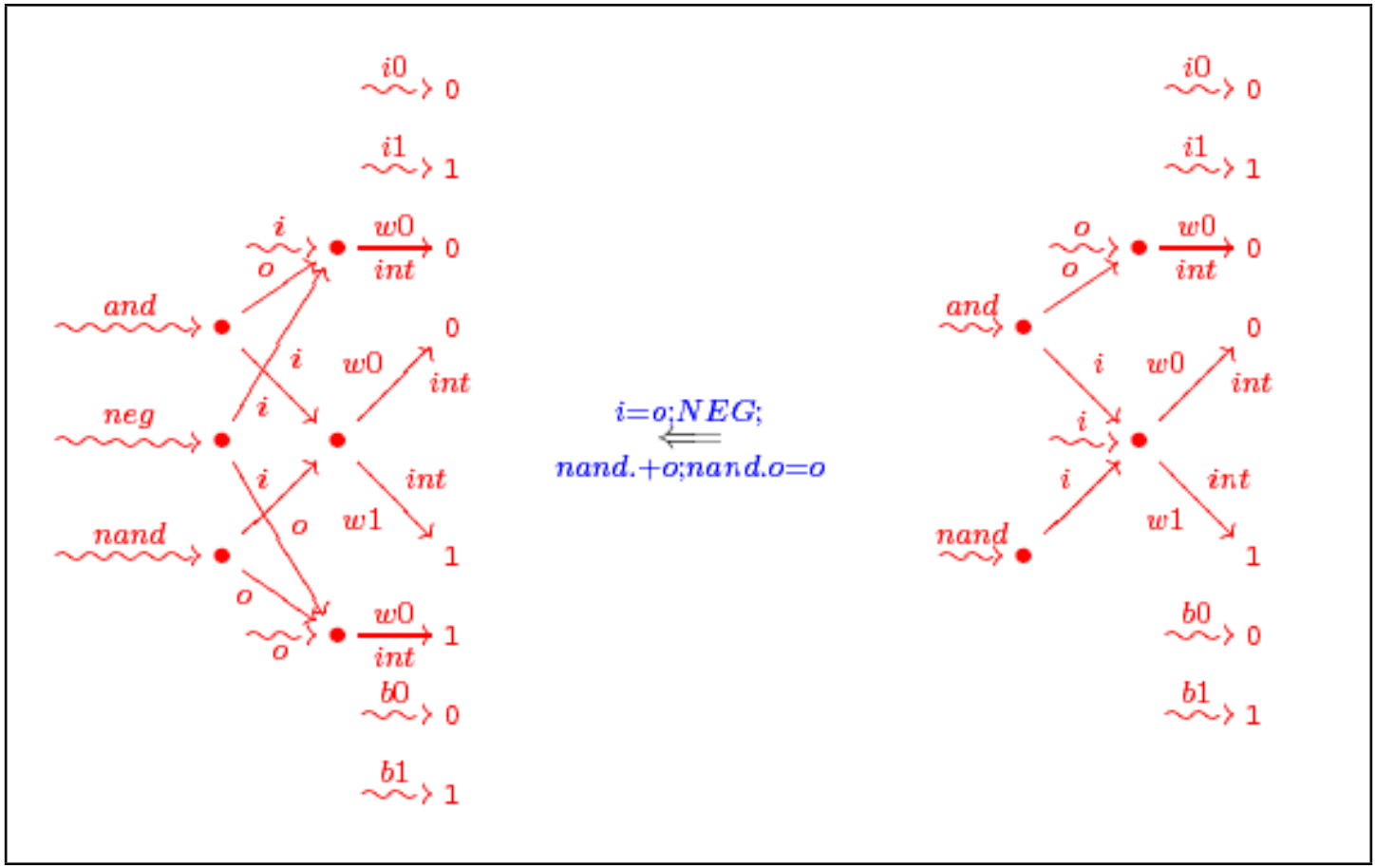
6.6 Rekenen met waarden



$\Downarrow AND$

6.6 Rekenen met waarden

$\Downarrow AND$

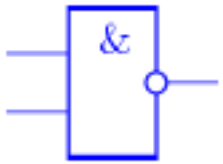


6.6 Rekenen met waarden

We zullen voor

$$NAND = nand = new; nand.{+}i; nand.i = i; AND; i = o; NEG; nand.{+}o; nand.o = o$$

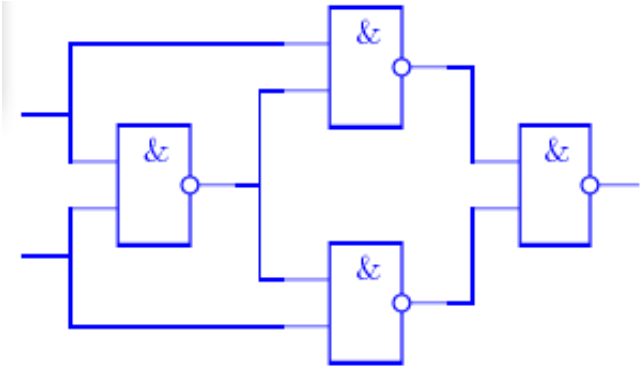
het symbool



gebruiken.

6.6 Rekenen met waarden

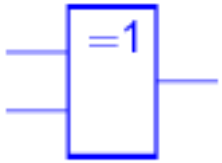
xor-poort:



$$
\begin{aligned}
XOR \quad = \quad & xor = new; xor.{+}i; xor.i = i; \\
& h0 = i.w0; h1 = i.w1; NAND; h2 = o.w0; \\
& I_{h0,h2}; NAND; h3 = o.w0; \\
& I_{h2,h1}; NAND; h4 = o.w0; \\
& I_{h3,h4}; NAND; xor.{+}o; xor.o = o
\end{aligned}
$$

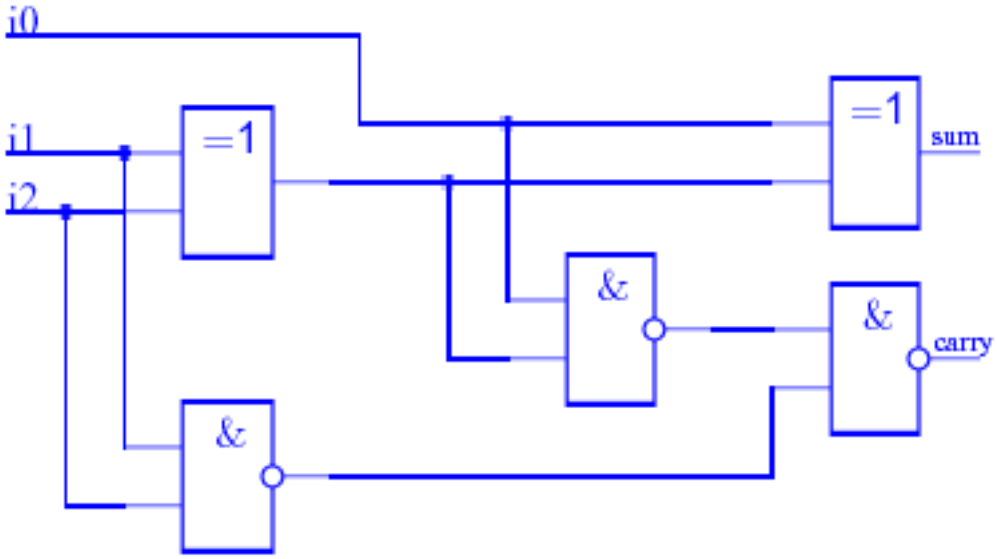
6.6 Rekenen met waarden

We zullen voor xor-poorten het symbool



gebruiken.

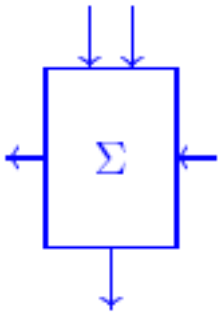
6.6 Rekenen met waarden



$$
\begin{aligned}
VOS \quad = \quad & vos = new; vos.{+}i; vos.i = i; a0 = i.w0; a1 = i.w1; a2 = i.w2; \\
& I_{a1,a2}; XOR; a3 = o.w0; I_{a0,a3}; XOR; vos.{+}sum; vos.sum = o; \\
& I_{a1,a2}; NAND; a4 = o.w0; I_{a0,a3}; NAND; a5 = o.w0; \\
& I_{a4,a5}; NAND; vos.{+}carry; vos.carry = o
\end{aligned}
$$

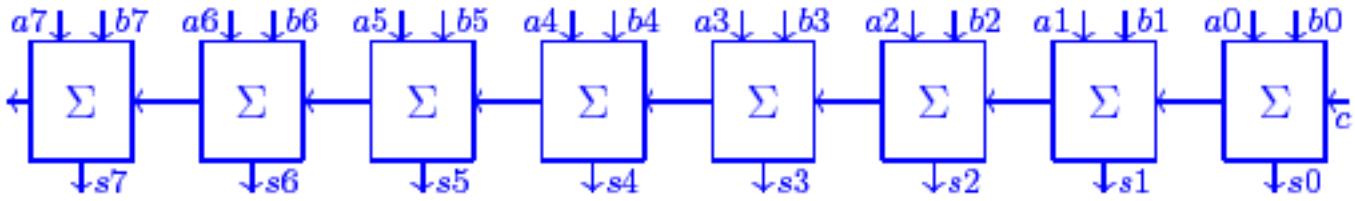
6.6 Rekenen met waarden

Voor een volledige optelschakeling zullen we het symbool



gebruiken.

6.6 Rekenen met waarden



$$
\begin{aligned}
8BITS \quad = \quad & 8bits = new; 8bits.{+}i; 8bits.i = i; \\
& I_{a0,b0,c}; VOS; 8bits.{+}s0; 8bits.s0 = sum; \\
& I_{a1,b1,carry}; VOS; 8bits.{+}s1; 8bits.s1 = sum; \\
& I_{a2,b2,carry}; VOS; 8bits.{+}s2; 8bits.s2 = sum; \\
& \vdots \\
& I_{a6,b6,carry}; VOS; 8bits.{+}s6; 8bits.s6 = sum; \\
& I_{a7,b7,carry}; VOS; 8bits.{+}s7; 8bits.s7 = sum
\end{aligned}
$$