

modelleren

I

In computers is informatie vaak gedefinieerd als relatie.

Initial set:

- empty set
- geen elementen

Als er een set is en we voegen er een element aan toe is het resultaat een set.

Als er een set is met 1 element en we verwijderen dat element is het resultaat de lege verzameling

- Twee dezelfde elementen maakt niet uit
- De volgorde van elementen maakt niet uit

Relatie is een subset van het Cartesisch product van verzamelingen.

$$\{A, B\} \downarrow \times \{C\} = \{(A, C), (B, C)\}$$

Unaire relatie: @ een kolom

Binare relatie: twee kolommen

Ternaire relatie: drie kolommen

II Universe of discourse: Het gedekte van de wereld wat door een model wordt beschreven.

Informele communicatie: lichaamstaal

Formele communicatie: geschreven tekst

→ alle wensen van de opdrachtgever komen in de "requirements document"

1. Splits alle informatie in elementaire sentences.
2. Groepen deze zinnen
3. Maak constantes voor de wissel waarden die steeds terugkomen
4. Gebruik standaardnamen voor alle entiteiten
5. Voeg layout toe zodat iedereen begrijpt waar het over gaat.

ORNF: Object van Normal form

v.b.: Cooperate with person Name "Janssen" works
for Department with DName "Sales"
→ constante ↓
 standaardnaam

↓

ORGR : object role grammar rule

1. Verwijder instanties
2. Schrijf standaardnamen tussen hoofdletters

v.b.: Cooperator (Name) works for Department
(DName).

Modellen:

[] : Label type

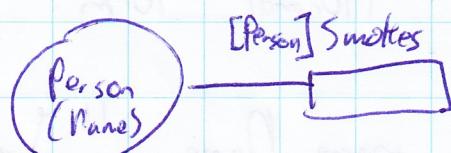
○ : Object type

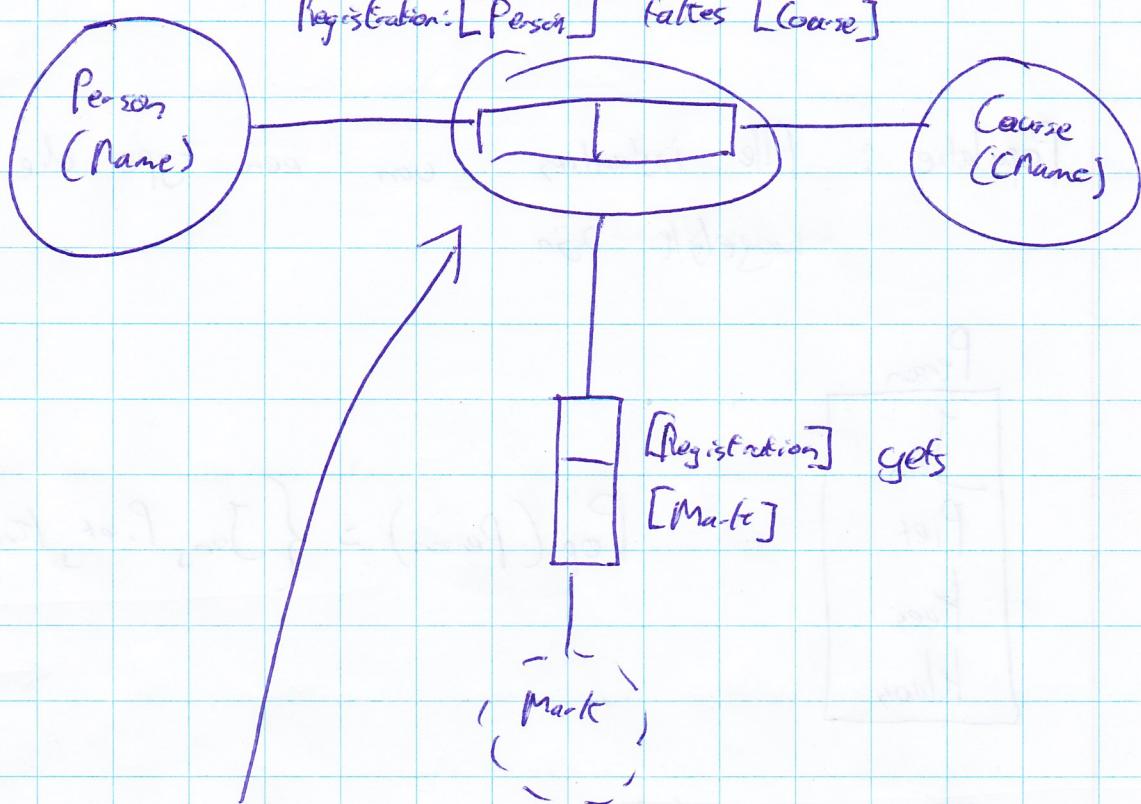
- [] : brugtype (tussen label en object)

- [] : feittype (tussen object en object)

- [] : Unair feittype

v.b.

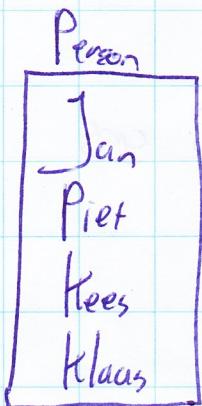




Objectificatie

III

Populatie : Alle instanties van een type die mogelijk zijn



$$\text{Pop}(\text{Person}) = \{\text{Jan}, \text{Piet}, \text{Kees}, \text{Klaas}\}$$

M = semantiek van een schema

M = set van alle mogelijke populaties in een schema

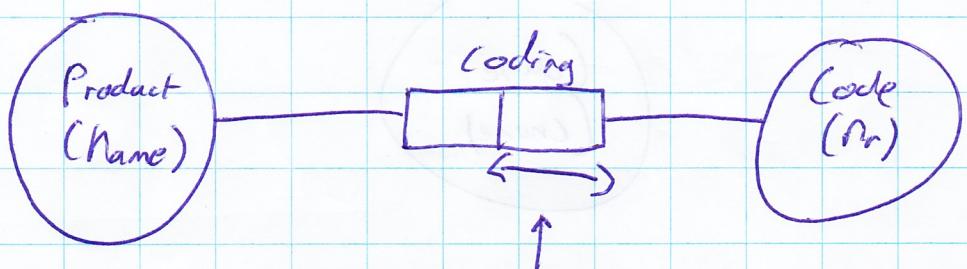
$$M(\Sigma) = \{P \mid \text{IsPop}(\Sigma, P)\}$$

Beperkingen (constraints) : nodig om alle fouten geworden uit te sluiten

1. Schema maken
2. Fouten uitsluiten

1. - Unique role constraint

→ pijl symbool boven een brug- of feritype.



Een code komt maar één keer voor

Populatie:

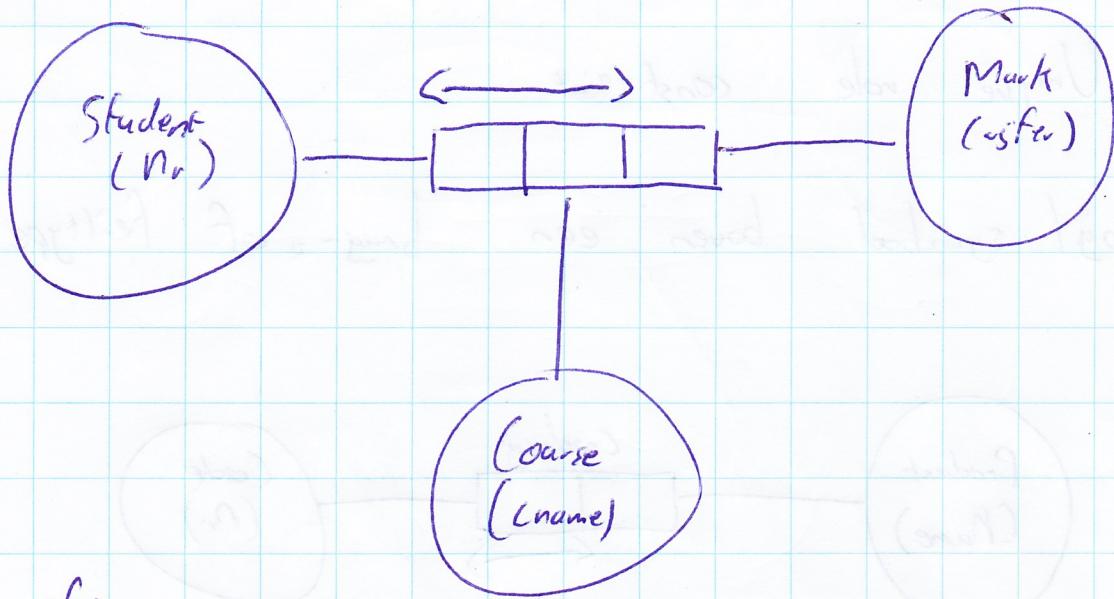
battery	1234
lump	1349
stelter	1234
batterij	1234
stelter	1376

may well

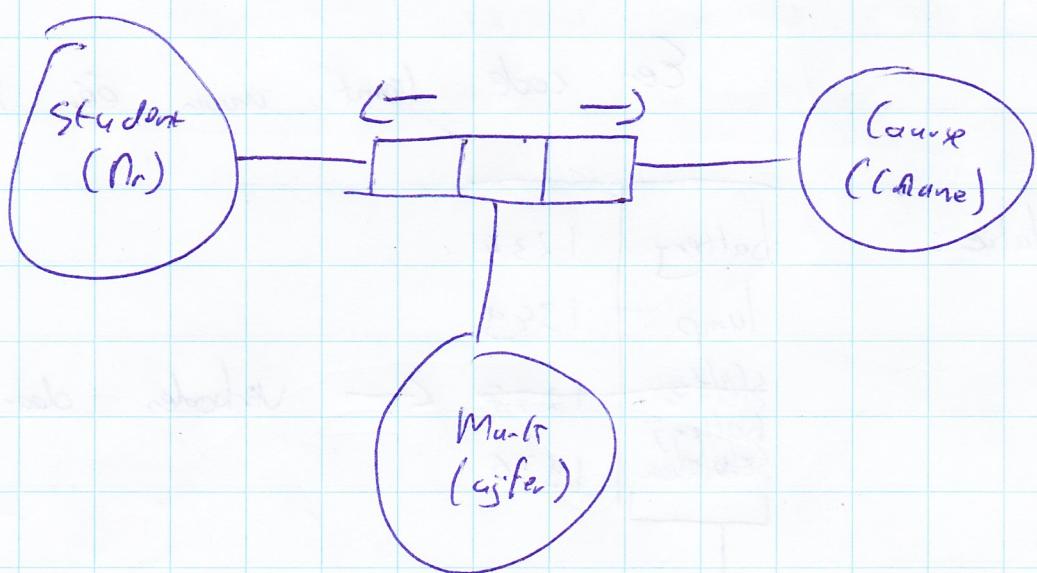
← verboden door de regel

- Multi role uniqueness

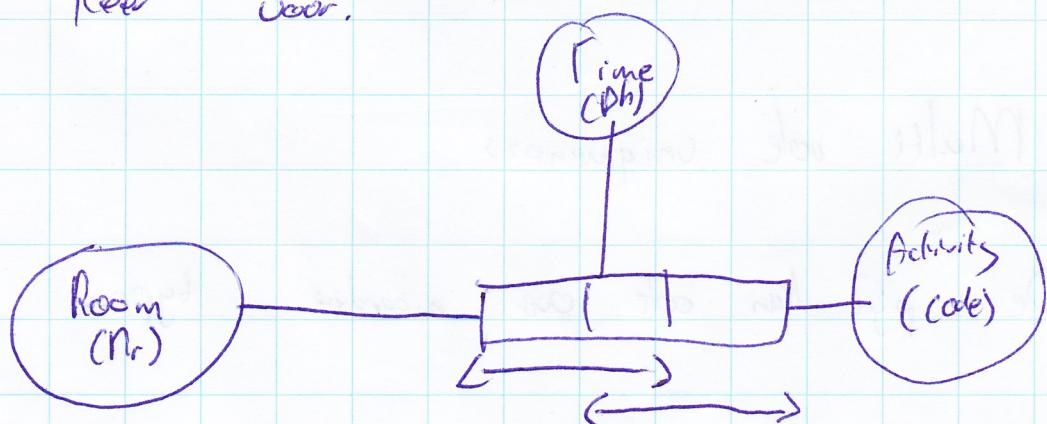
→ De pijl kan ook over meerdere typen



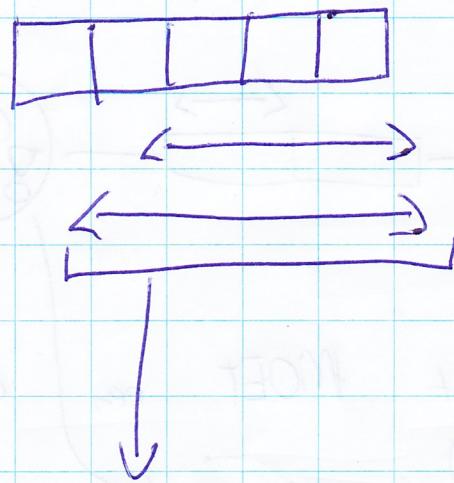
of:



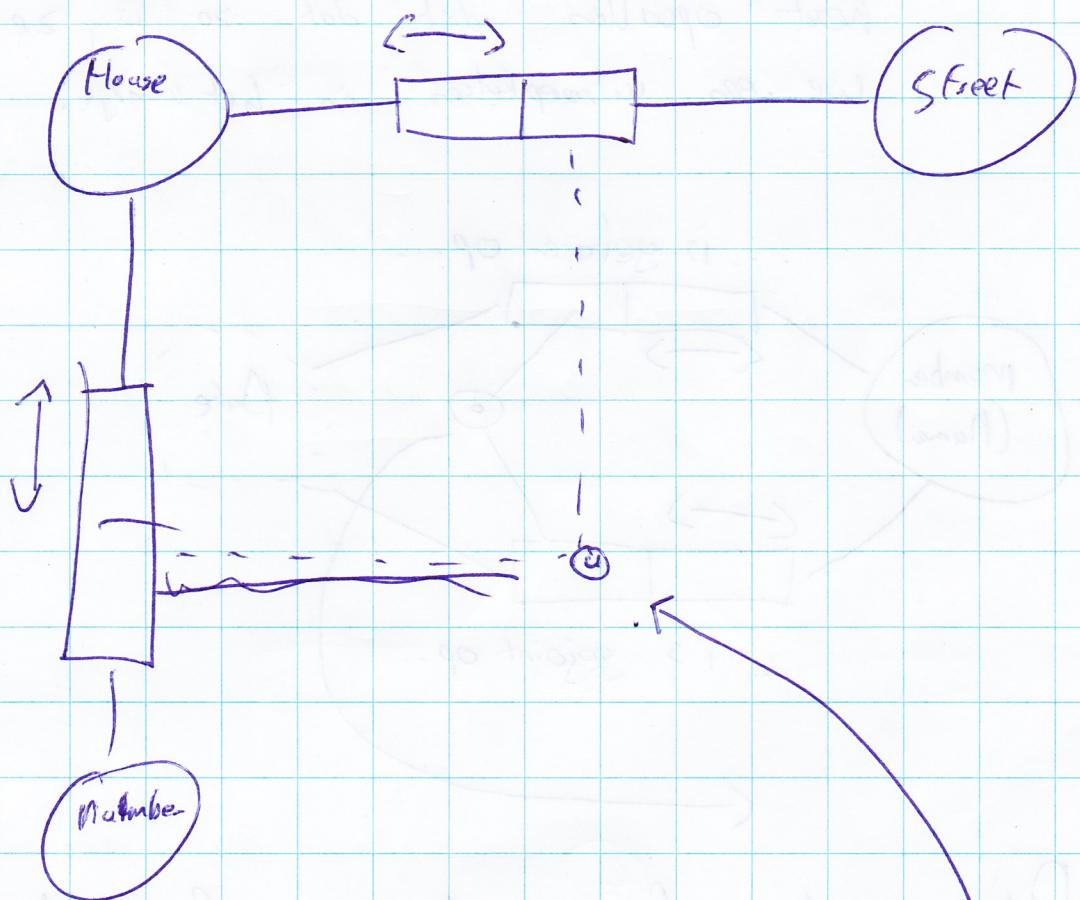
Elke combinatie Student - Course komt voort
één keer voor.



Meerdere constraints kan opleggen.

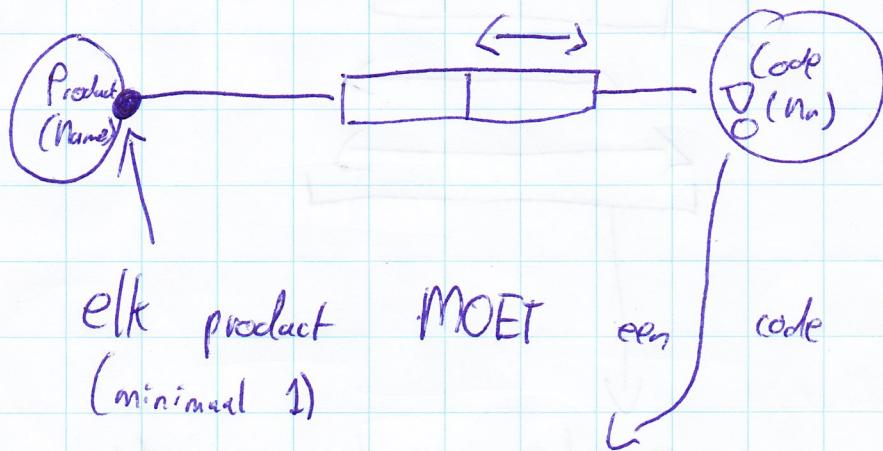


Nuttelos es kann weggedeuten werden



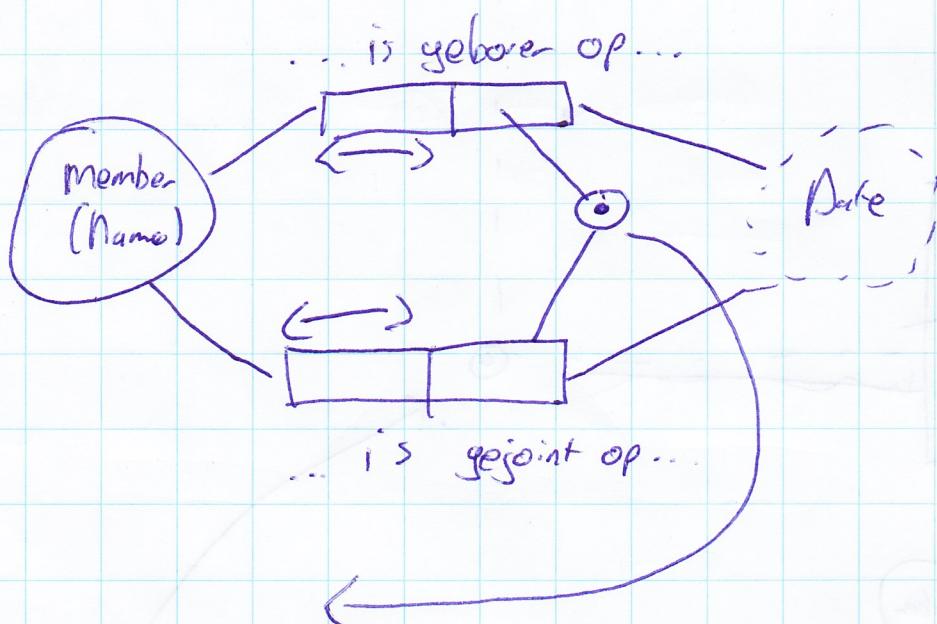
Kun ook over verschillende feittypen

2. Total role constraints



elk product MOET een
(minimaal 1) code hebben

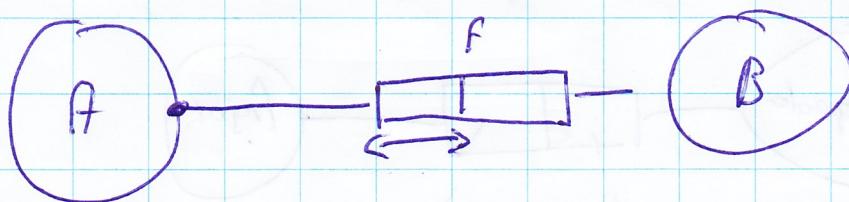
Als een object geen • heeft en het moet oppellen dat dat zo is, zetten we een uitroepteken in het rondje.



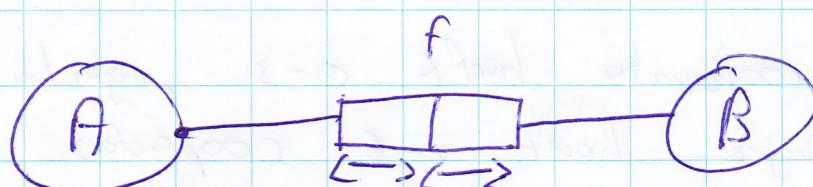
Date moet of in de ene, of in de ander voorkomen (alleheue mag ooit).

Functie:

$$f: A \rightarrow B$$



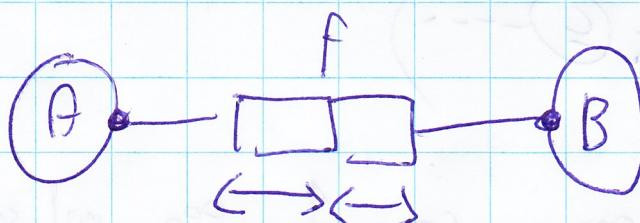
$$f: A \rightarrow B \quad (\text{injectief, one-to-one})$$



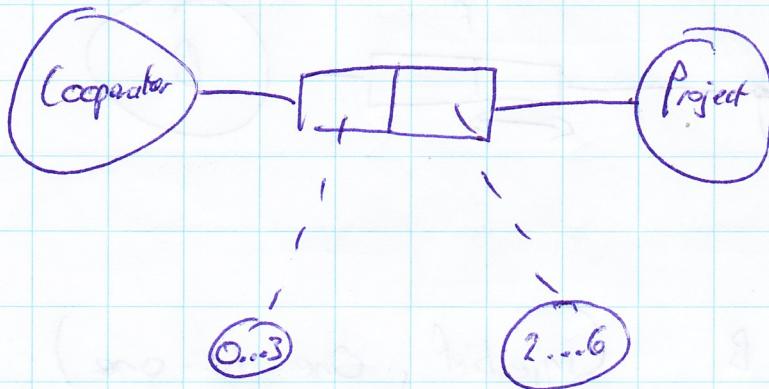
$$f: A \rightarrow B \quad (\text{surjectief, onto})$$



$$f: A \rightarrow B \quad (\text{bijection})$$

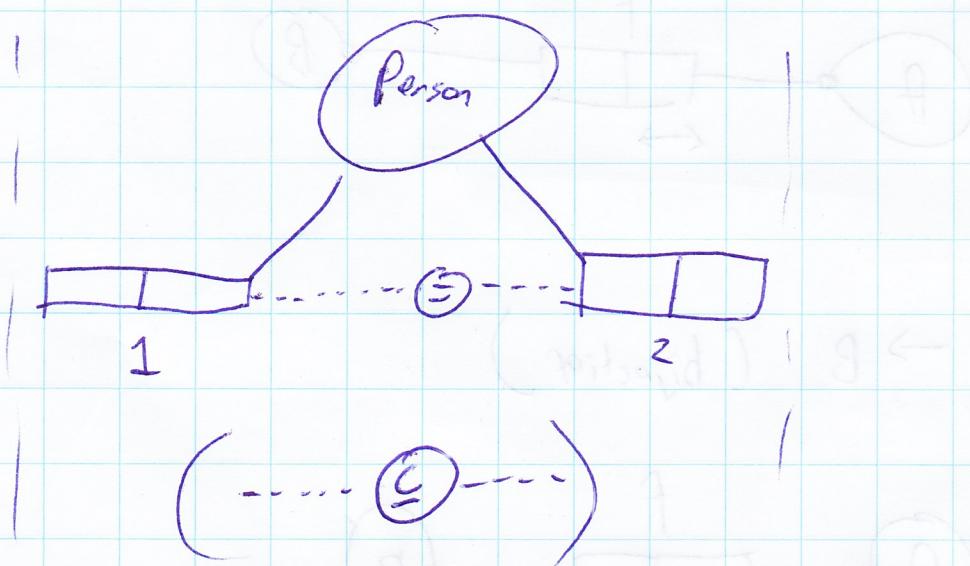


3. Frequency constraints

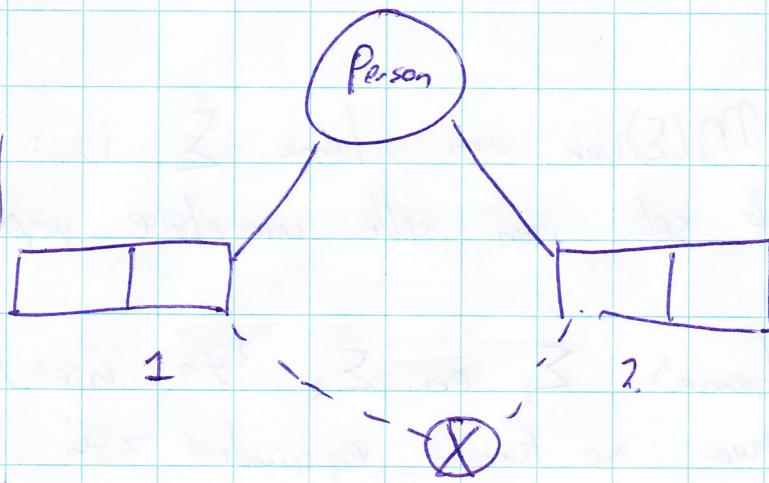


- Elke cooperator heeft 0-3 projecten
- Elk project heeft 1-6 cooperators

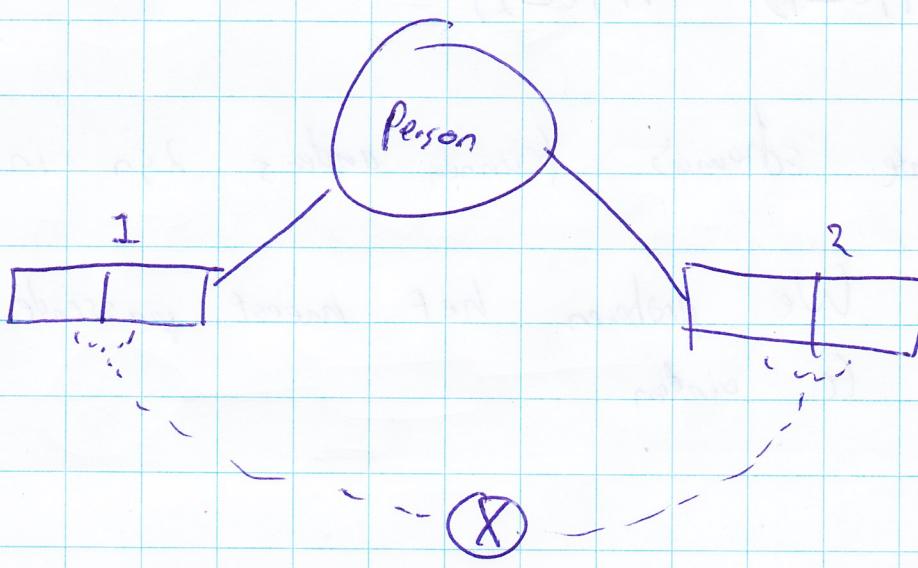
4. Heterogeneous constraints



Als een persoon in 1 een nul speelt,
dan speelt de persoon in 2 ook een nul
en andersom.



Person speelt een rol in of 1 of 2. of geen.



De combi 1 of de combi 2 bestaat.

1. Meaning $M(\Sigma)$ van een schema Σ is:
de set van alle mogelijke populaties
2. Twee schema's Σ_1 en Σ_2 zijn niet hetzelfde
maar ze kunnen equivalent zijn
3. Twee schema's Σ_1 en Σ_2 zijn equivalent als
ze dezelfde betekenis hebben:
 $m(\Sigma_1) = m(\Sigma_2)$
4. Equivalent schema's kunnen anders zijn in gebruik
We proberen het meest passende schema
te vinden

IV

Grammatica: Een grammatica beschrijft zinnen die toegestaan zijn in talen.

↓
beschrijft zinnen

→ malen van zinnen
→ begrijpen van zinnen

Syntaxis: beschrijft de structuur van de zin

Semantiek: beschrijft de betekenis van de zin

Pragmatiek: beschrijft de intentie (wanneer de zin wordt gebruikt).

Een zin is een welgewomde opeenvolging van woorden.

Lexicon: Alle woorden die ingevuld ^{mogen} kunnen worden door de grammatica. → meestal alle label woorden

Syntactische categorieën beschrijven de structurele elementen van een zin

Voorbeeld:

sentence: subject, verb, object.
↑ ↑ ↗
defined as followed by end of rule

subject:

"I".
"you". ← separator of alternatives

verb:

"amuse".
"call". ← woorden uit het lexicon

object:

"you".
"the man".

→ "I call you" "you amuse the man"

Een zin kan ook parameters hebben:

Sentence (PERSON, NUMBER)



Sentence (PERSON, NUMBER) : subject (PERSON, NUMBER),
verb (... , ...)

Parameter waarden staan in een ander soort regel.

PERSON :: FIRST | SECOND | THIRD.

NUMBER :: SINGULAR | PLURAL.

↑
defined as

↑
separator

↑
end of rule

Subject singular:

subject (FIRST SINGULAR) : "I".

subject (SECOND, SINGULAR) : "you".

subject (THIRD, SINGULAR) : "he", "she", "it".

en2.

...

verb (FIRST | SECOND, SINGULAR | PLURAL): "walt".
verb (THIRD, SINGULAR): "waltis".

Phz. ...

Parameters:

name :: variable | variable | ... | variable.

↓

Zin:

Zin (name) : woord (name), @ woords (name).

↓

Elementen:

woord (variable_1) : "boot".

woord (variable_2) : "auto"; "bo".

woord (variable_3 | variable_4) : "toein".

Grammatica in modelleren:

Cooperator:

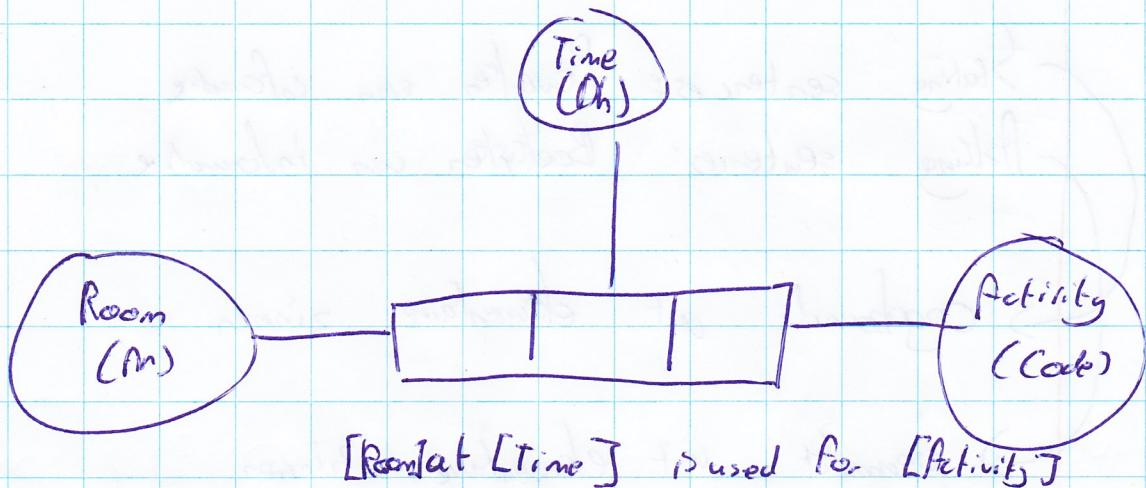
"Cooperator with" Name.

Name:

"Name", string.

→ Cooperator with Name Gaußen

Ternair feittype:



Reservation : Room, "at", Time, "is used for", Activity.

Room: "Room with" Nr.

Time: "Time with" Dh.

Activity: "Activity with" Code.

Nr: "Nr" string.

Dh: "Dh" string.

Code: "Code" string

(Commands) (Queries)

Stating sentences: Bewerken van informatie
 Asking sentences: Bevragen van informatie

→ opgebouwd uit elementaire zinnen

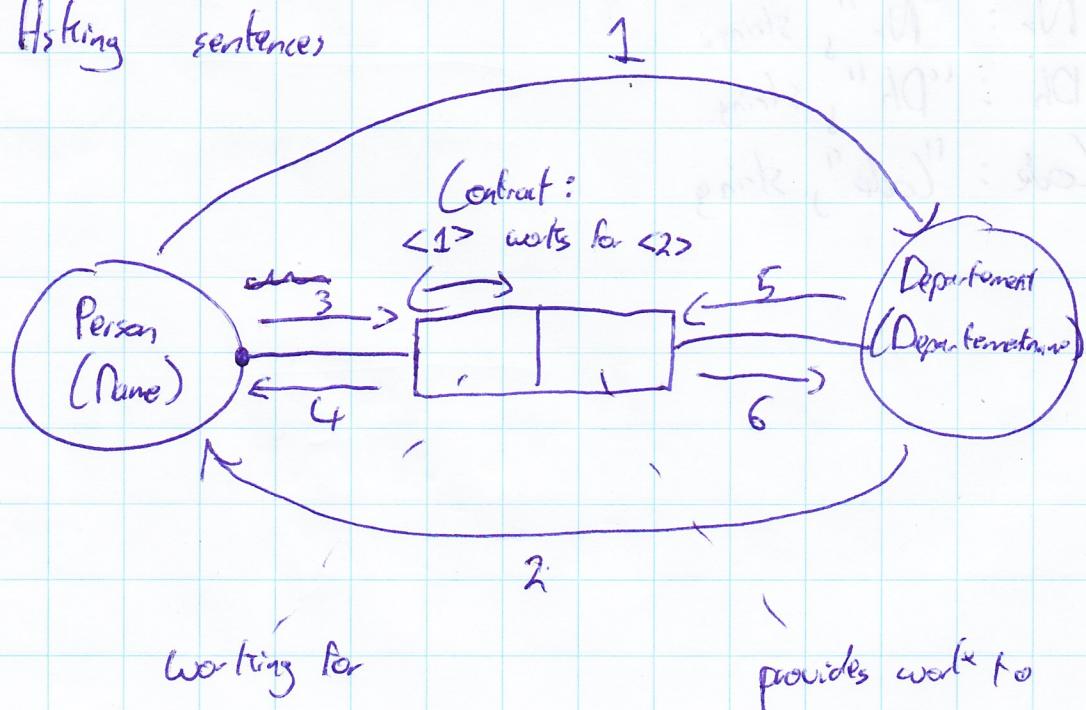
→ gevormd uit elementaire zinnen

→ gebruikt constructors om nieuwe zinnen te vormen

Ub.

Stating sentence:
 "ADD Person with Name "Jansson" works for Department with Departmentname "Sales"."

Asking sentences



1. Person working for Department

Person - Department relation

2. Department provides work to Person

Department - Person relation

3. Person in Contract

Person - Contract relation

4. Contract from Person

Contract - Person relation

5. Department has Contract

Department - Contract relation

6. Contract for Department

Contract - Department relation

Astking sentence Expressions

LIST : Laat alle instanties van een object zien

Vb LIST President



President with Name " ... "

...

Rod

of = object type

rn = role name

lv = Label value

OT - RN : Als heb je een lijst wil hebben van
objecten die in een bepaalde rol
verkennen.

Vb President having spouse.



President	Spouse
A	①
B	2
C	3
...	...

↓
LIST

...

↗

President with Name "A".

QT - RN - OT αA : Een relatie laten zien.

vb Person being spouse of President

Spouse	Person	President
1		A
2		A
3		B
...		...

LIST

Person with Name "1".

DISTINCT: Twee dezelfde output's verwijderen

vb DISTINCT President having spouse.

A
B
...

VB: Om op een bepaald label value te zoeken
gebruiken we een LU.

OT - RN - OT - RN - OT - LU

Person being spouse of President with Name "A"

↓

1

2

— — — — — — — —

Q: COUNT: Telt het aantal feiten op vanuit
de informatie die gegeven is.

VB COUNT President.

↓

39

— — — — — — — —

AVERAGE: neemt het gemiddelde van het
eerst genoemde objecttype

VB AVERAGE Age being death age of President.

↓

64

— — — — — — — —

MAXIMUM : laat de hoogste waarde van het eerstgenoemde objecttype zien

vb. MAXIMUM Age being death age of President.

↓

83

— — — — — — —

SUM : tellt het aantal eerste genoemde objecttype bij elkaar op

vb. SUM Nr. of children resulting from Marriage.

↓

0138

— — — — — — —

GROUPWISSE : neemt het laatste object (na of) en
groepert daarop.

vb. COUNT GROUPWISSE President being a member of Party.

↓

4

X

2

Y

...

↙

↑

AND ALSO: combineert twee condities

BUT NOT: eerste conditie moet waar zijn, tweede moet onwaar zijn.

OR OTHERWISE: Of de eerste, of de tweede conditie moet waar zijn.

THAT: Als een dezelfde variabele & het twee voor hetzelfde moet zijn.

(Relatie tussen het begin en het eind van de query.)

ANOTHER: Als een variabele anders moet zijn dan de eerder genoemde waarde.

Meerdere waarden uit de resultaten halen:

FROM

ub. Name of,

Year being birth year of,

Name of Person being spouse of

FROM

President having spouse.

Vb Name of

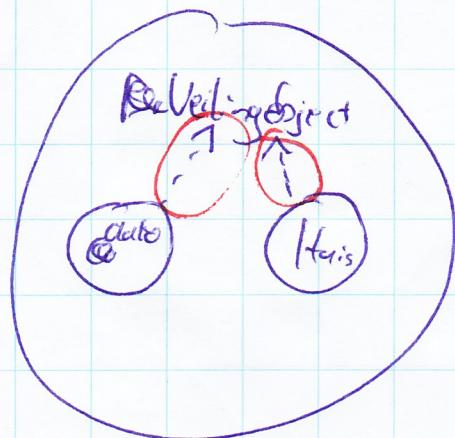
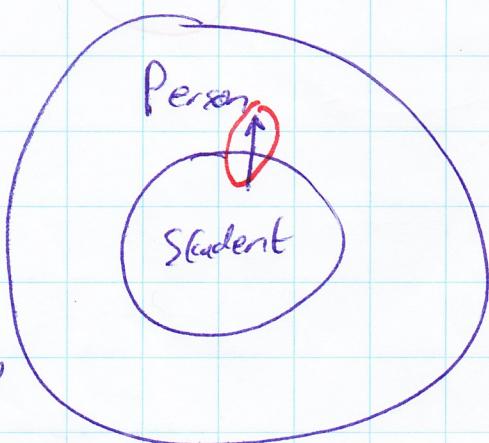
Year being birth year of

From

President having served No of years

20

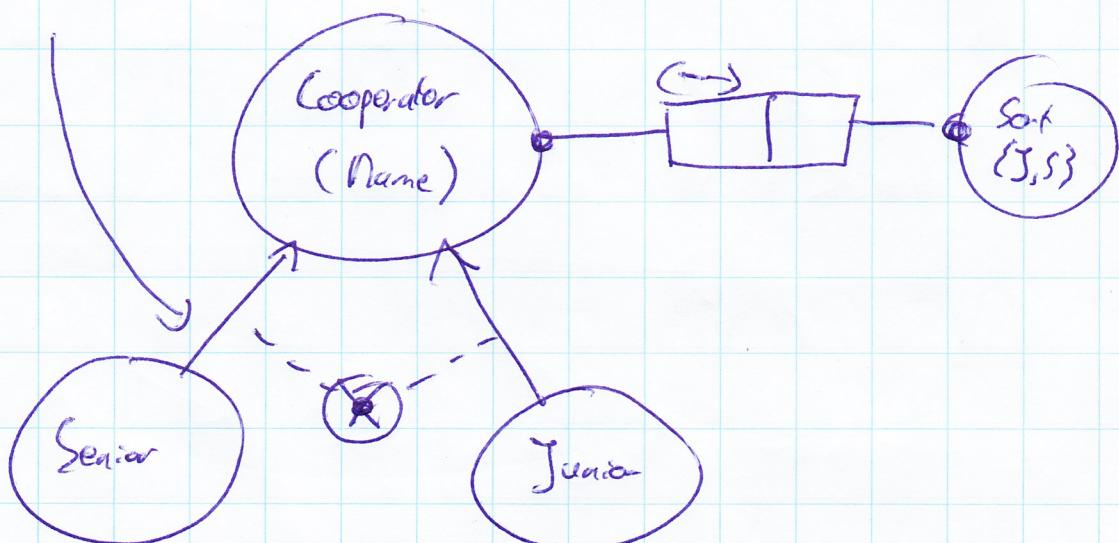
Om dubbele feittypes te voorkomen kunnen we hierarchie aanbrengen



Identiteit: ↓

Properties : ↗ ↓

Specialisatie:



ORG:

Senior

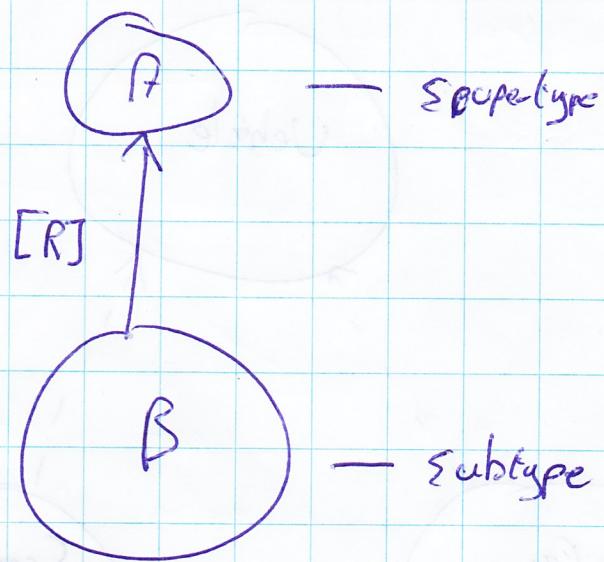
Junior

IS Cooperator with

IS Cooperator with

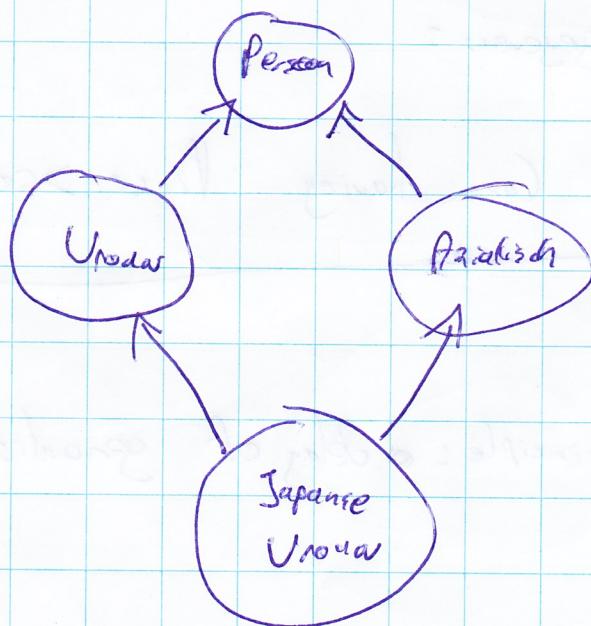
Sort "S"
Sort "J".

Als B een generalisatie is van A :



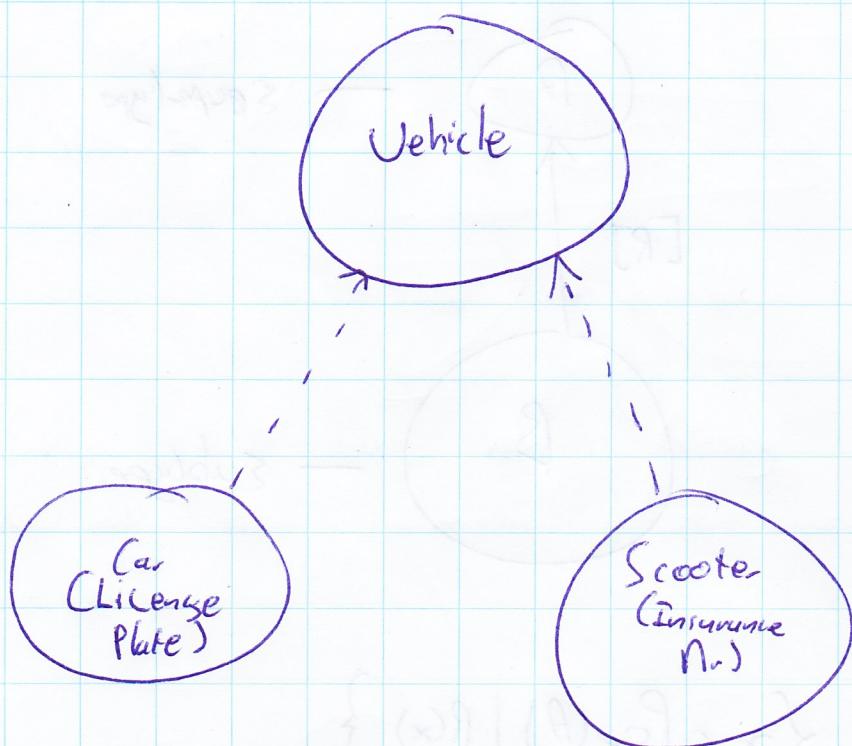
$$\text{Pop}(B) = \{x \in \text{Pop}(A) \mid R(x)\}$$

Een generalisatie kan ook naar meerdere supertypen gaan:



Voorwaarde: De supertypen komen uit bij hetzelfde (ze) supertype.

Generalisatie:



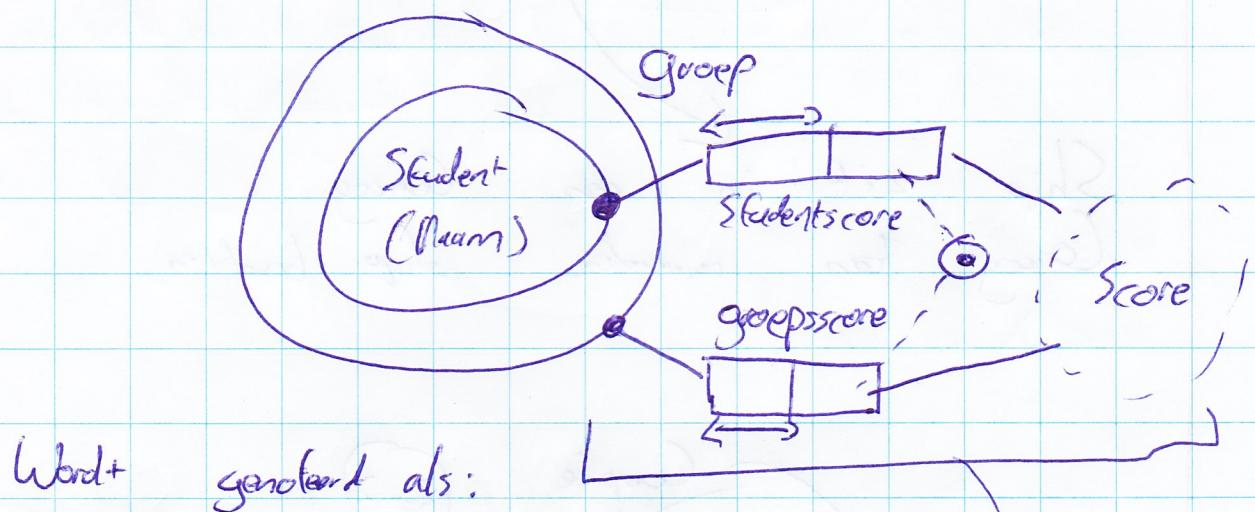
Vehicle generaliseerd. car, scooter

Ophalen van gegevens:

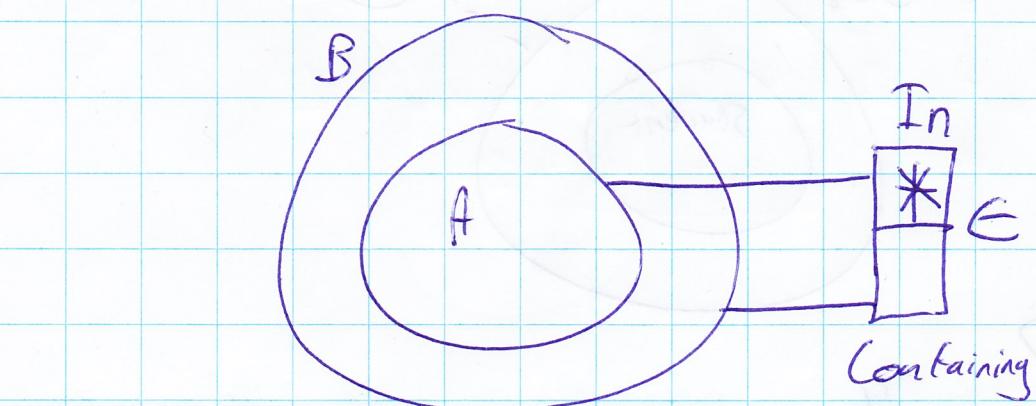
License plate of car having Price > 5000

principle: o delay of generalisation imp.

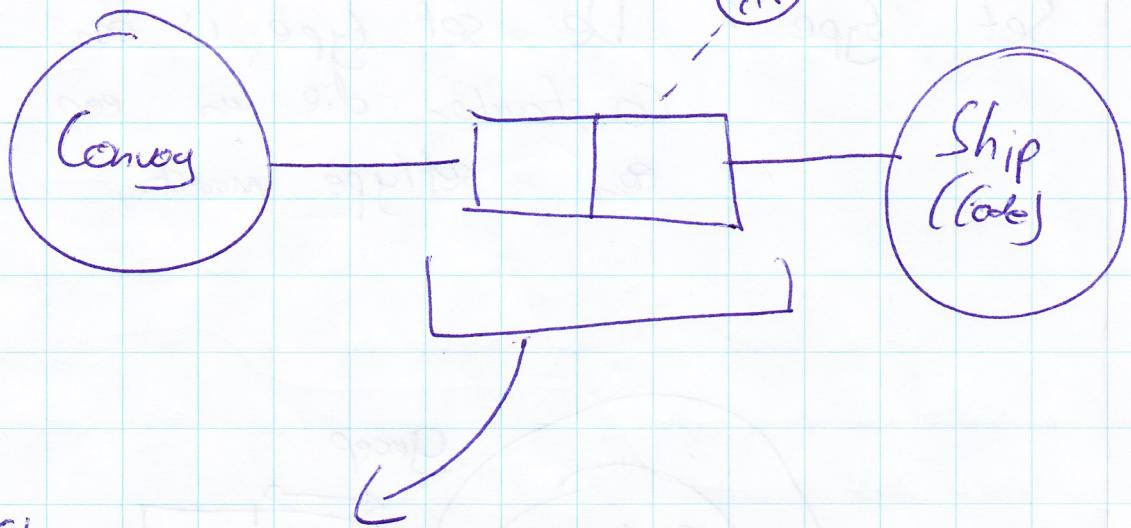
Set type : De set type is een speciale constructor die van een basistype een settype maakt.



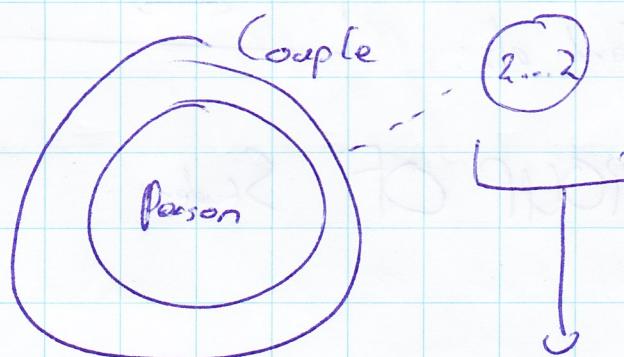
Score assigned to Group
CONTAINING Student.



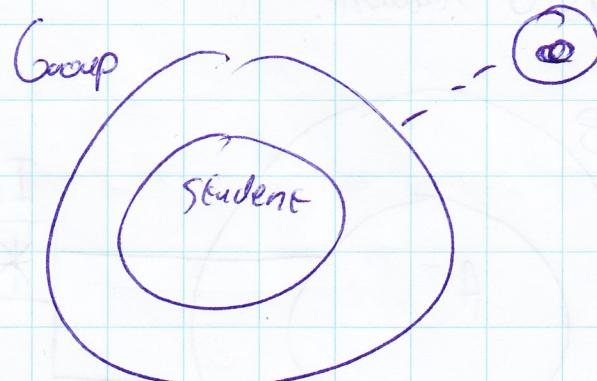
Tussen het basistype A en het settype B bestaat een feittype E. Dit feittype is impliciet (niet nodig om te tekenen).



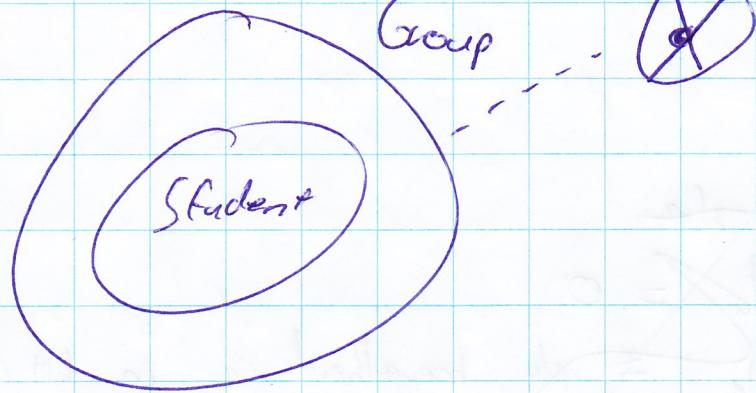
Ship zit in een
Convoy kan meerdere ships berichten



2 mensen in een
couple.



alle studenten zitten in een
groep.



- D Alle studenten zitten in één groep
 O (Elke student speelt max 1 keer de rol)

