Towards An Ontology of Computer Software

Speaker: Xiaowei Wang

Supervisor: John Mylopoulos, Nicola Guarino

University of Trento, Italy
ISTC-CNR, Italy
xwang@disi.unitn.it

What is software ?

- Algorithm (e.g. a bubble sorting algorithm)
- Source code (e.g. encoded in Java/C)
- Realization of source code (e.g. the code stored on a hard disk)
- Running process of algorithm (e.g. sorting process running in a computer)

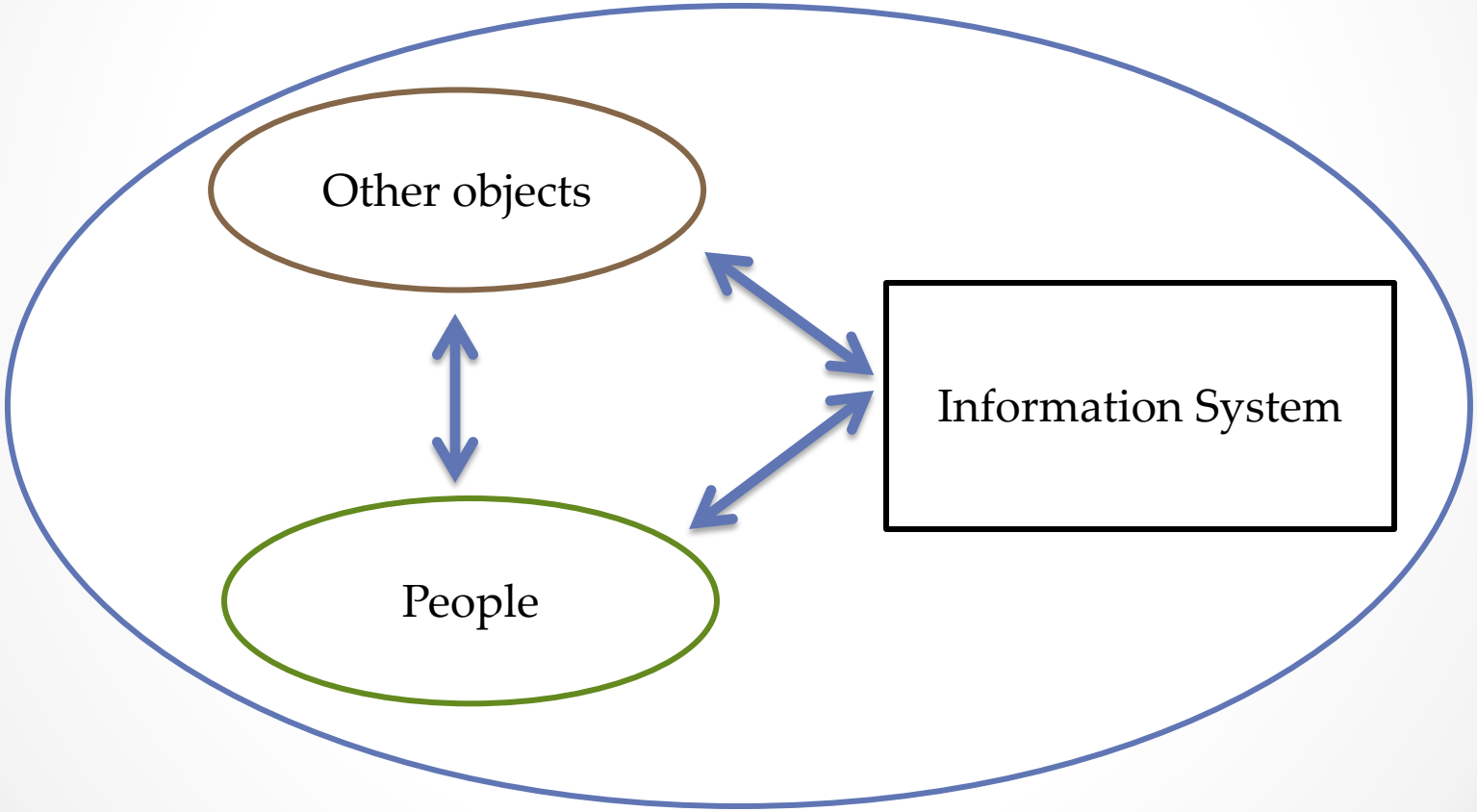
- Specification document?
- Design document?
- Requirement?

We try to provide a ***unified concept*** of software.

Definition of computer software

Computer software is an ***artifact*** consists of ***computer programs*** that implements a ***protocol*** (created from a ***specification***) in order to satisfy some ***requirements*** under some ***domain assumption***.

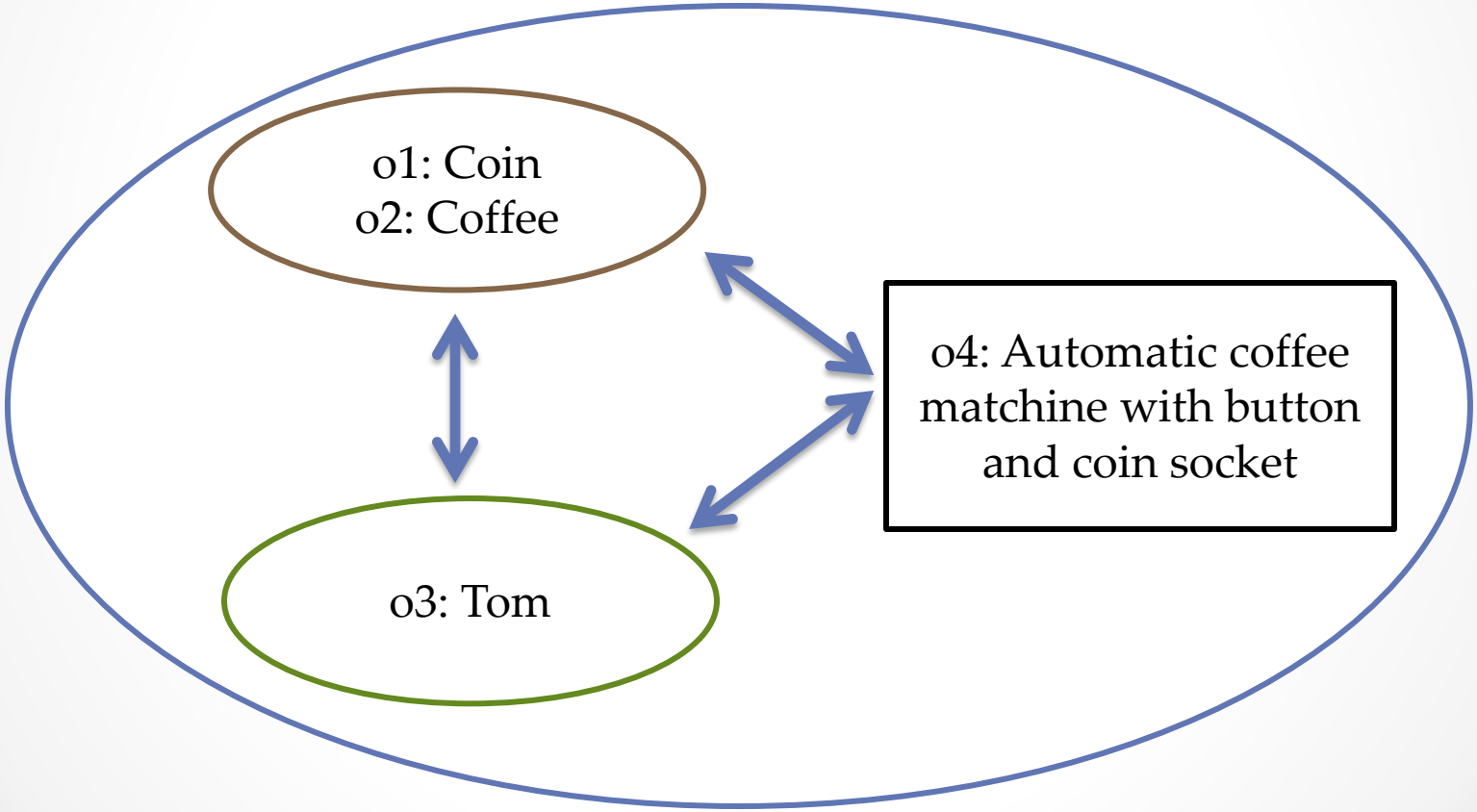
Social-technical system



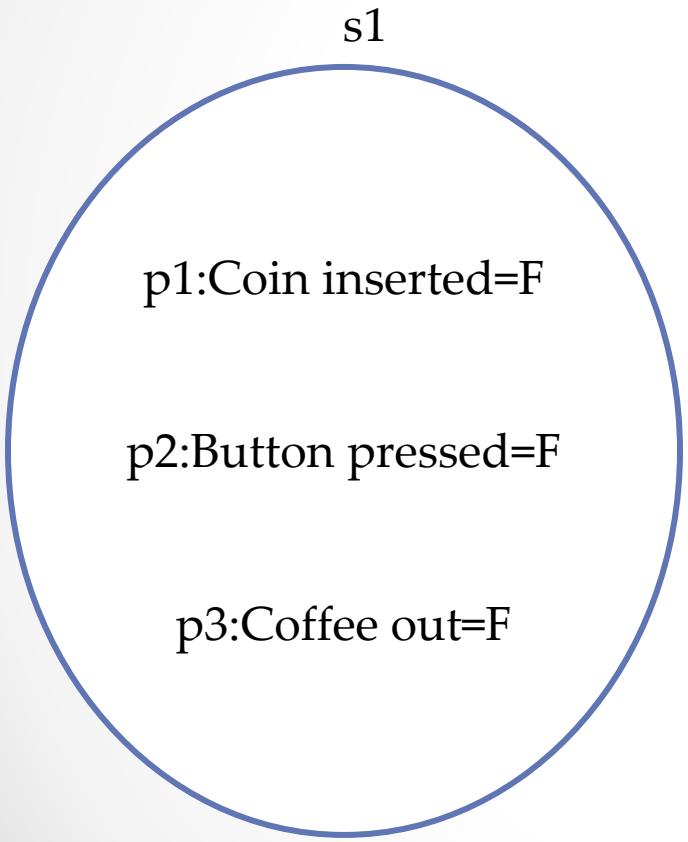
Coffee machine example

Tom

Socialtechnical system



State of affairs



p1: describes the fact if the coin is inserted or not

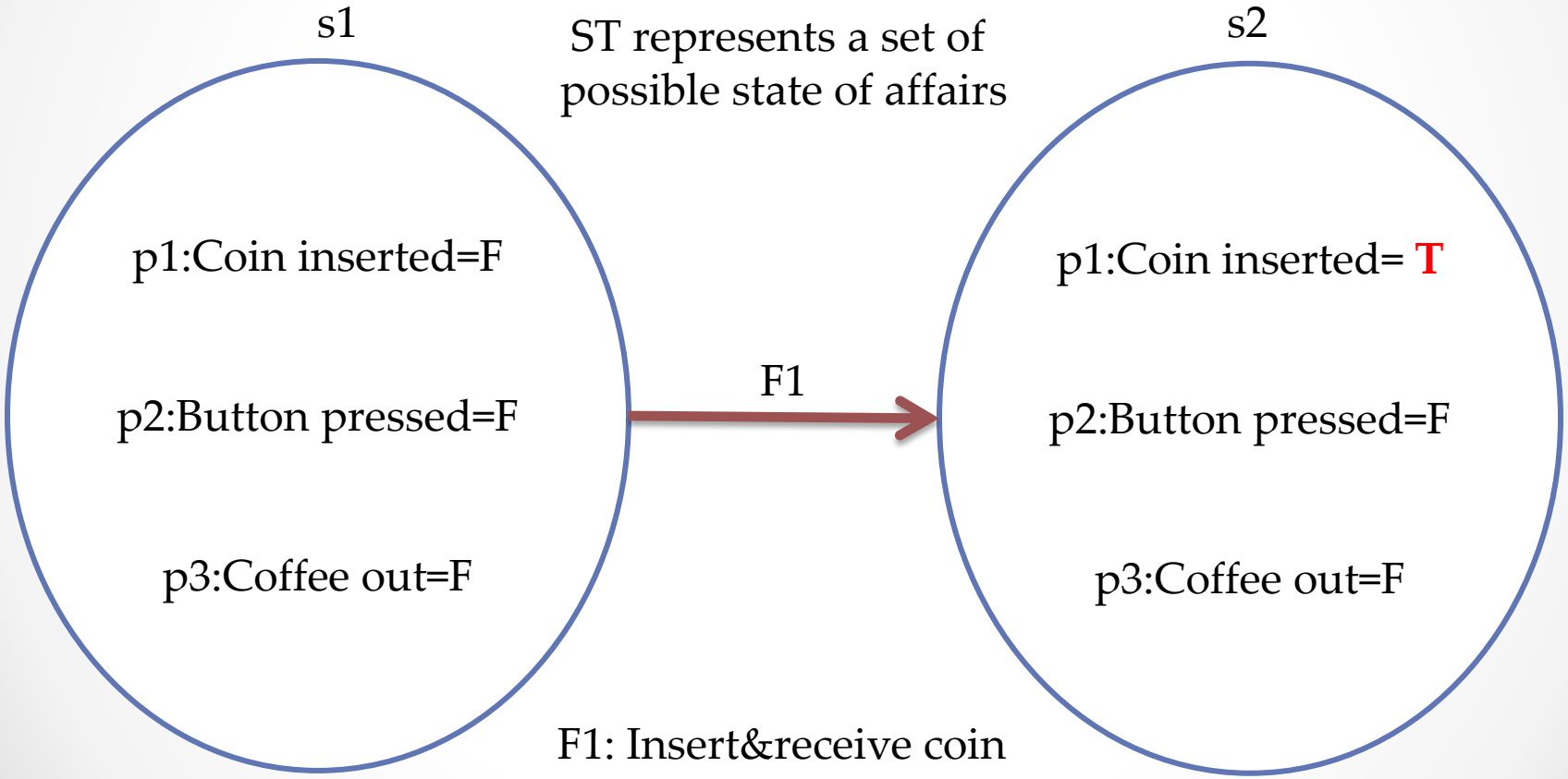
p2: describes the fact if the button is pressed or not

p3: describes the fact if the coffee is given out or not

Function

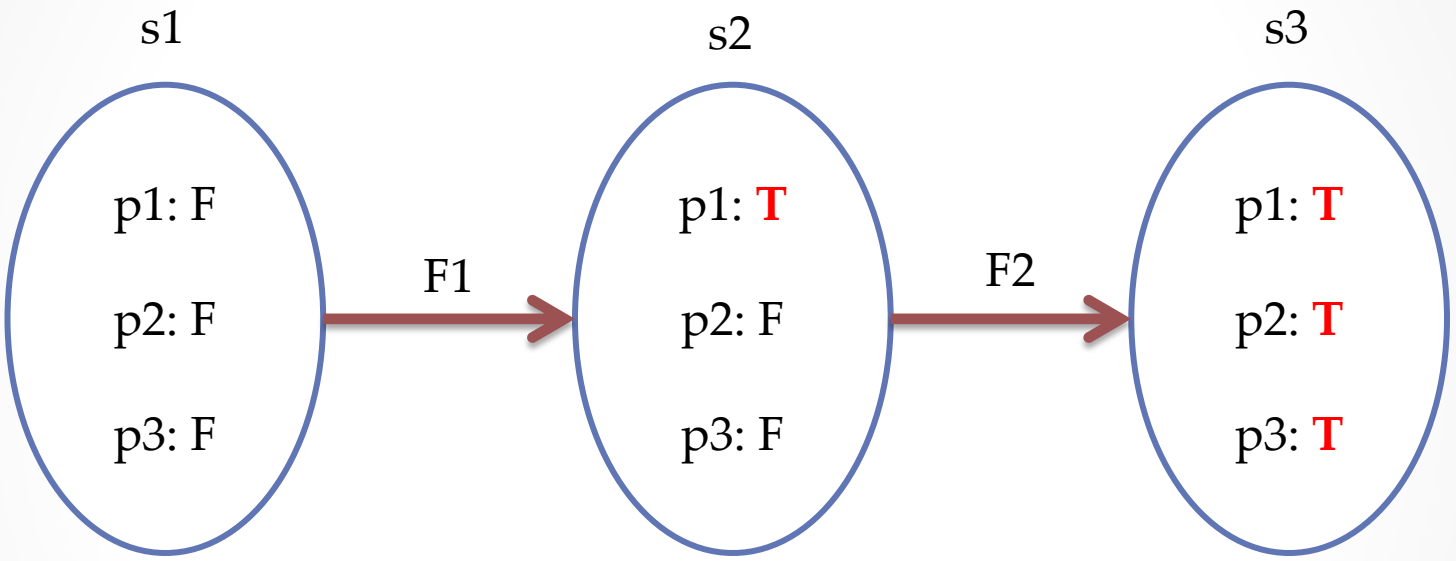
$f: ST \rightarrow ST$

ST represents a set of possible state of affairs



F1: Insert&receive coin

Protocol

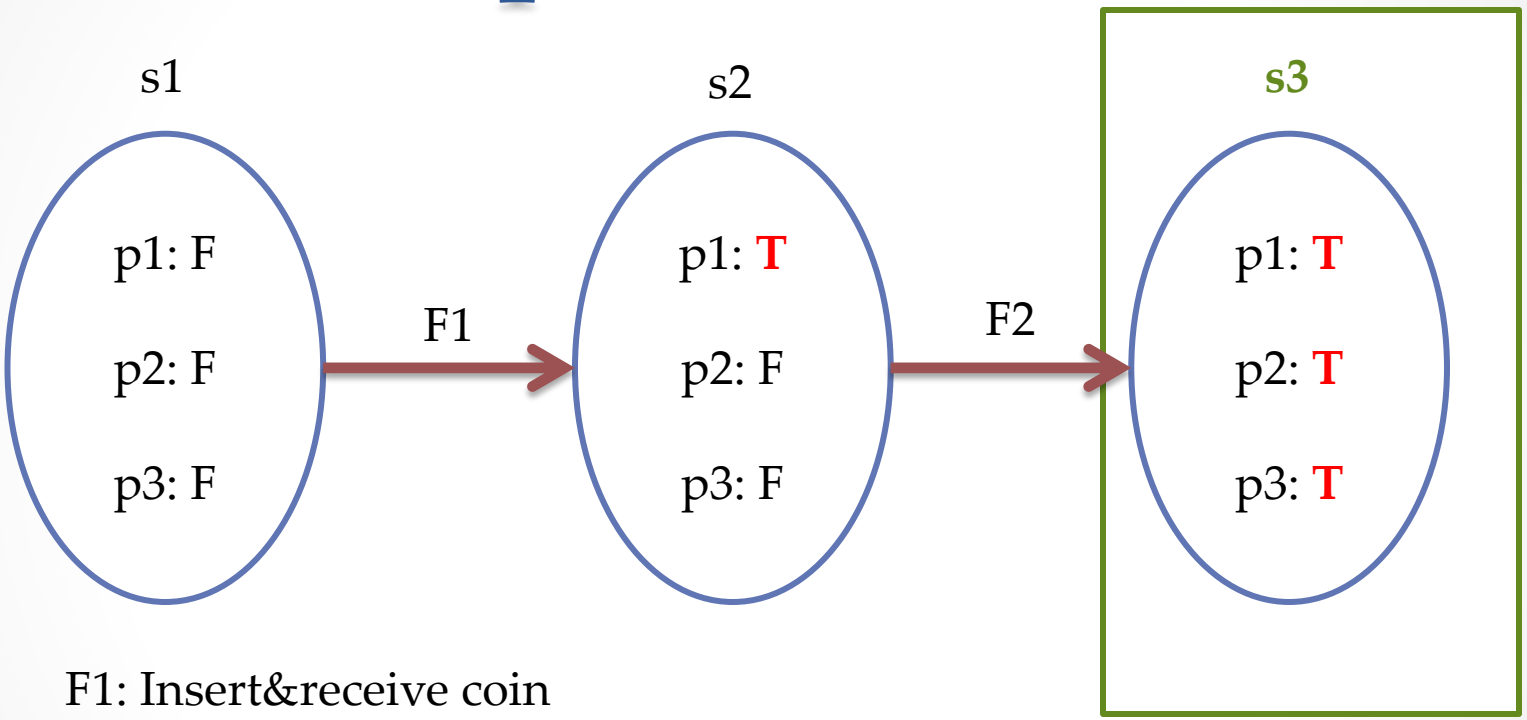


F1: Insert&receive coin

F2: Button press&coffee give out

P1: <F1,F2> **P2: <F2,F1>**

Requirement



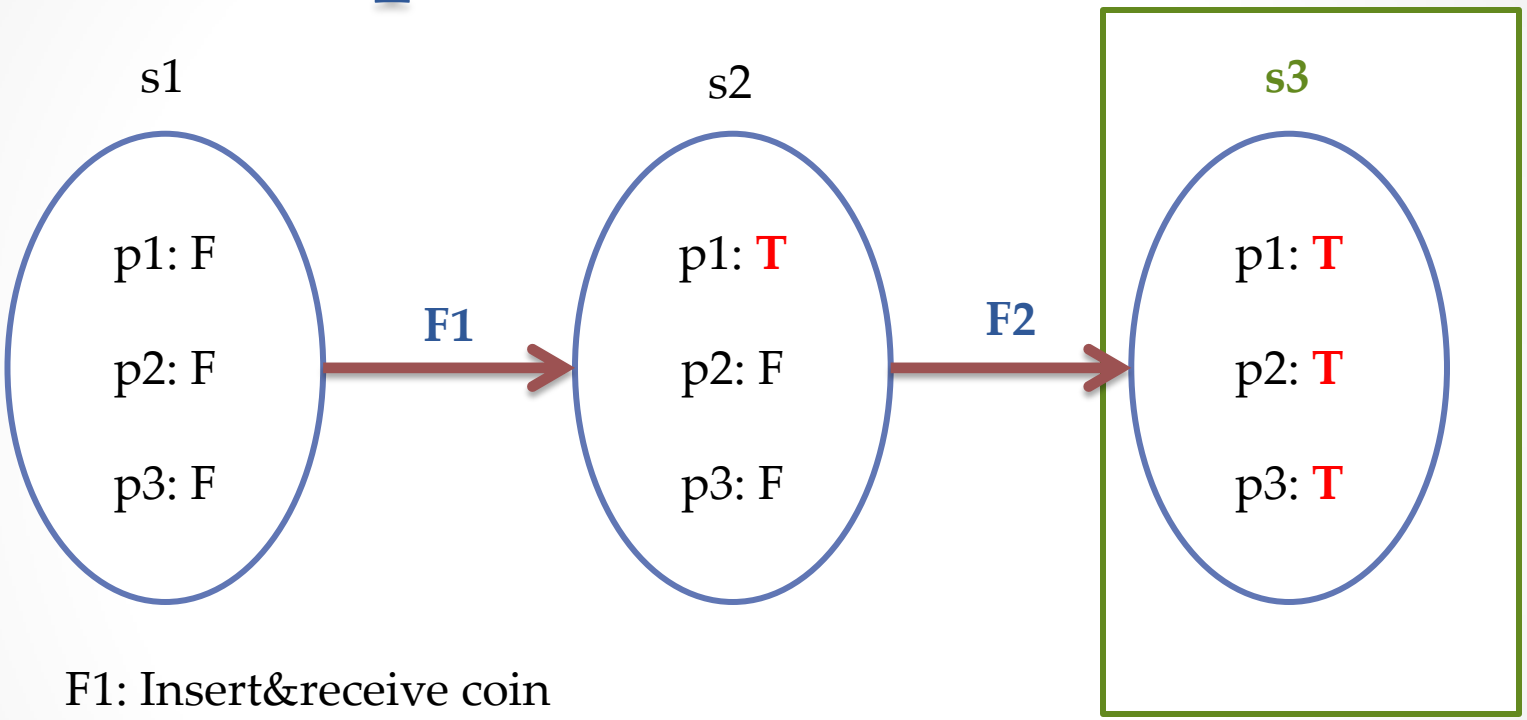
F1: Insert&receive coin

F2: Button press&coffee give out

P1: <F1,F2>

R:{s3}

Specification



F1: Insert&receive coin

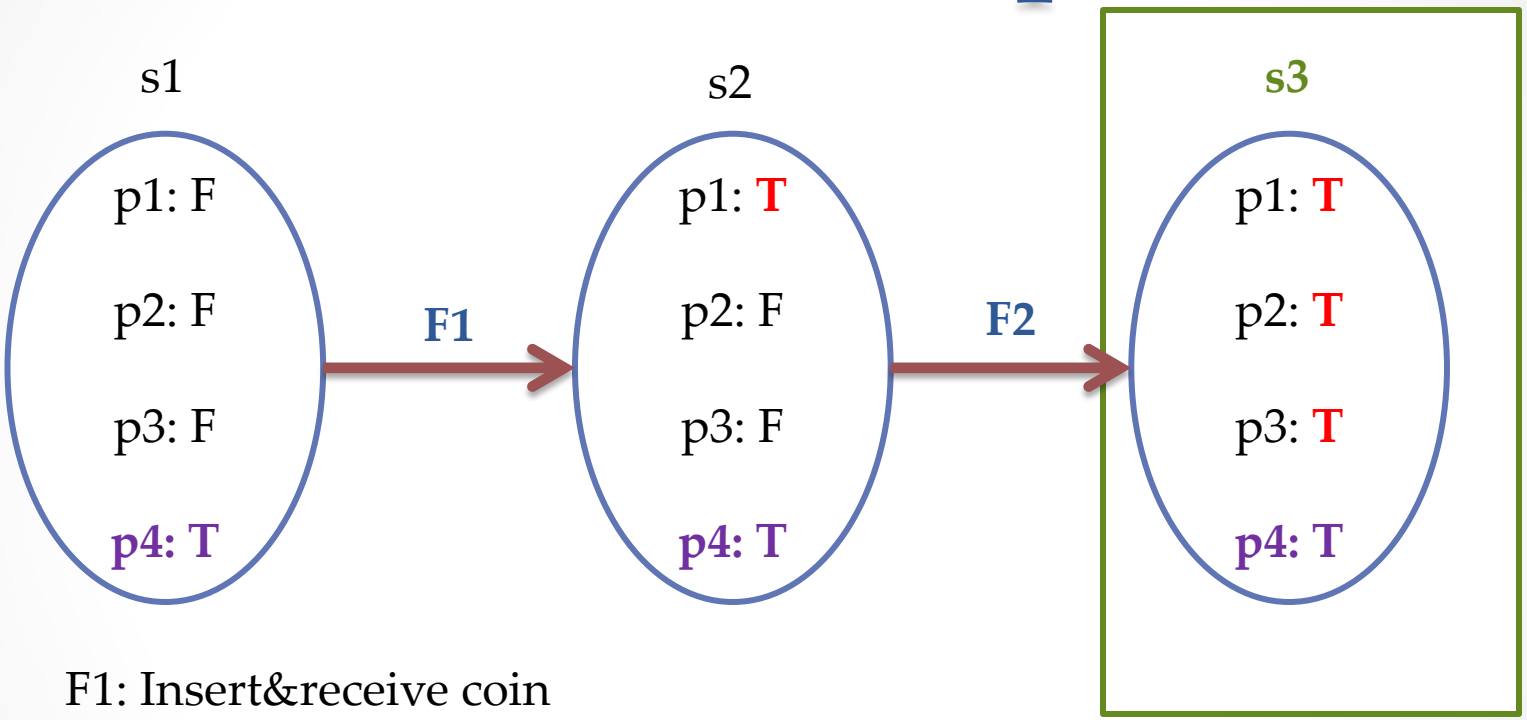
F2: Button press&coffee give out

P1: <F1,F2>

R:{s3}

S: {F1,F2}

Domain assumption



F1: Insert&receive coin

F2: Button press&coffee give out

P1: <F1,F2>

S: {F1,F2}

R:{s3}

D: {p4:T}

Put them all together

Propositions for the facts in the sate of affairs:
p1: describes the fact if the coin is inserted or not
p2: describes the fact if the button is pressed or not
p3: describes the fact if the coffee is given out or not
p4: power to the coffee machin is on

State of affairs (ST):
ST={s1, s2, s3}
s1={p1:F, p2:F, p3:F, p4:T}
s2={p1:T, p2:F, p3:F, p4:T}
s3={p1:T, p2:T, p3:T, p4:T}

Domain assumption (D):
D={p4:T}

Requirement (R):
R={s3}

Functions:
F1: Insert&receive coin
F2: Button press&coffee give out

Specification (S):
S={F1,F2}

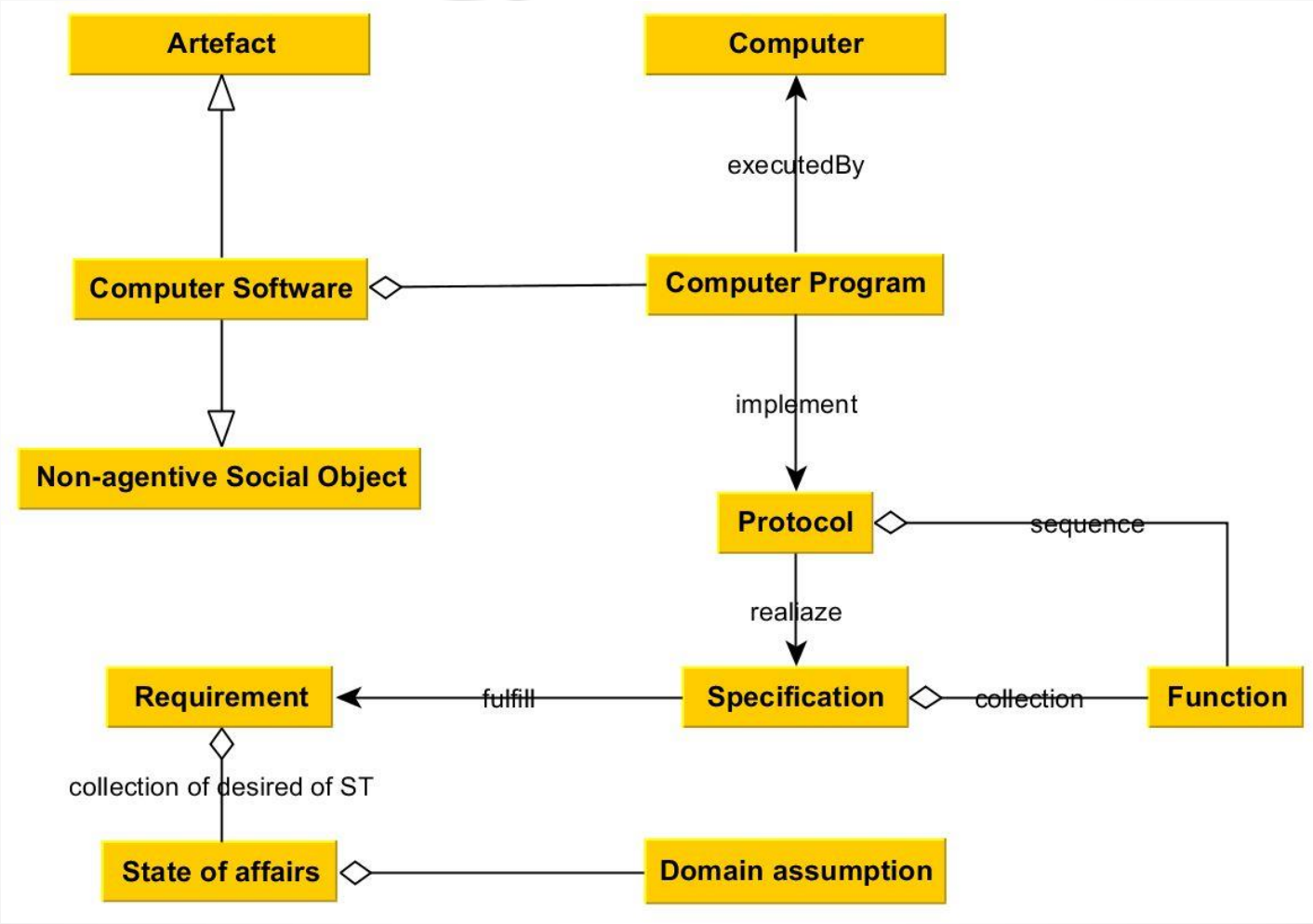
Protocol (P):
P=<F1,F2>

Formula expression

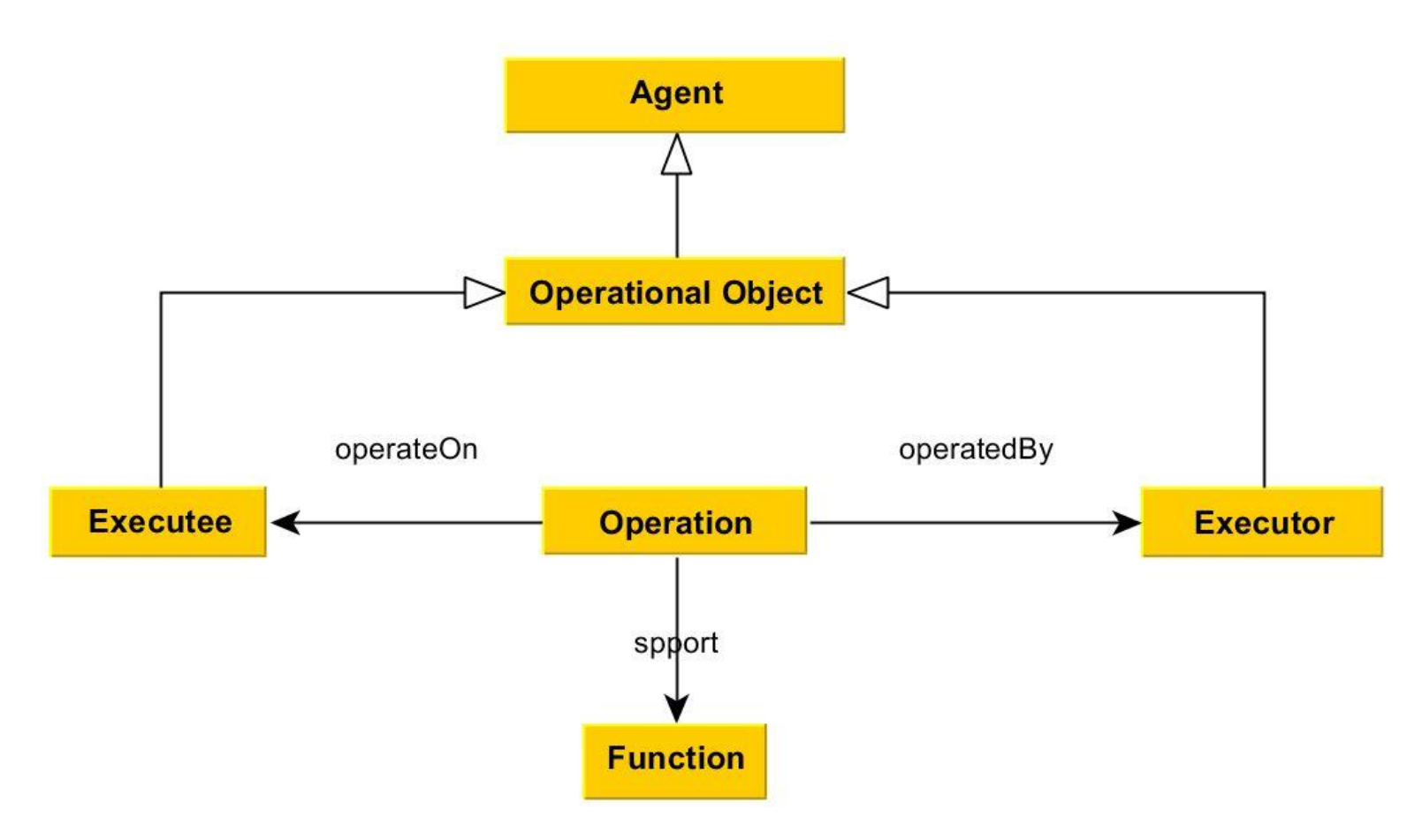
- $\exists S, R.\ \text{fullfil}(S, R) \rightarrow \exists P.\ \text{realize}(P, S)$

- $\exists P.\ \text{realize}(P, S) \rightarrow$
  $[\forall \text{event}.\ \text{execute}(\text{event}, P)$
  $\wedge\ \text{holds}(D, \text{time}(\text{event})) \rightarrow \text{post}(\text{event}) \in R]$

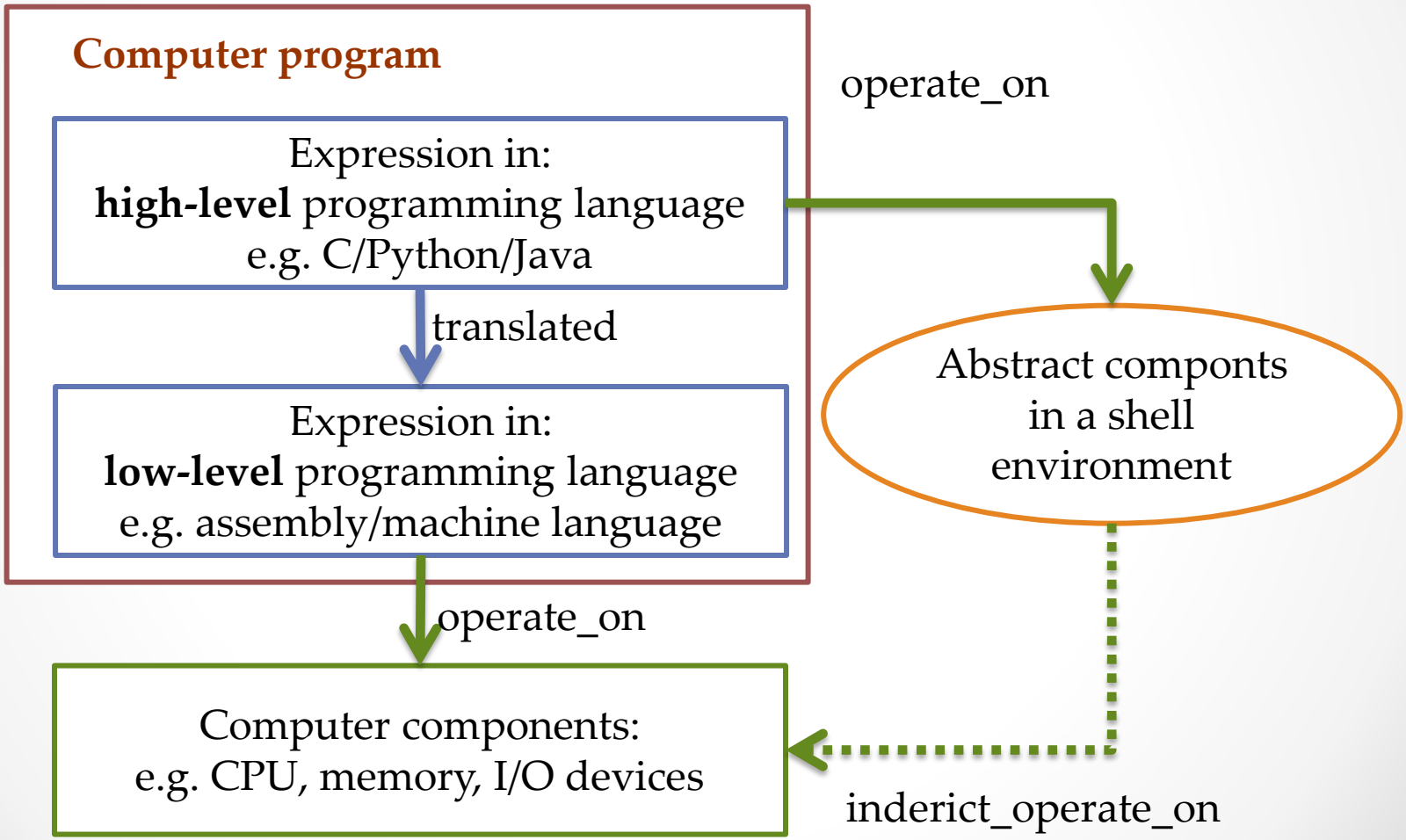
Ontology of software



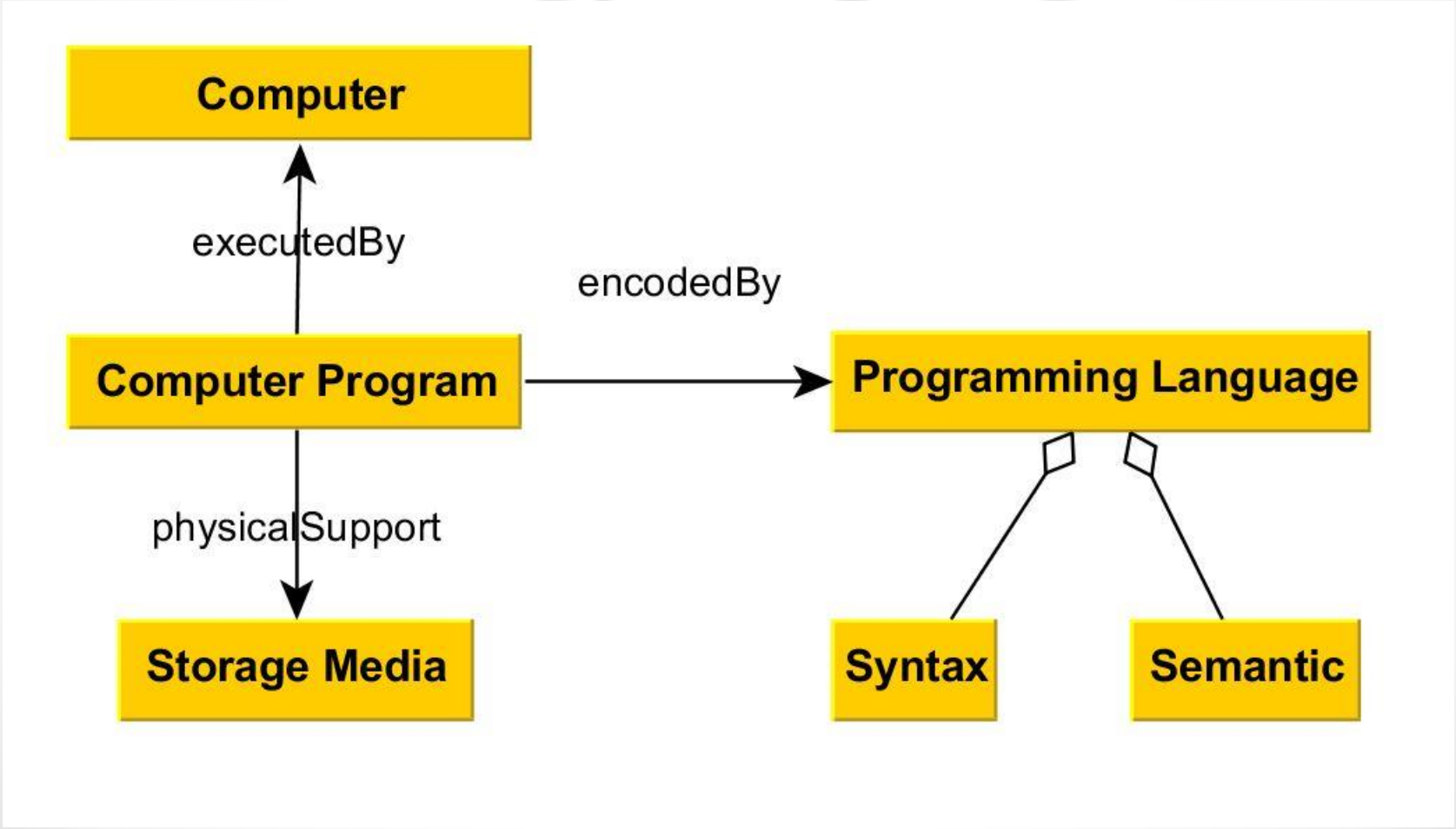
Operation underlying a function



Computer program



Ontology of program



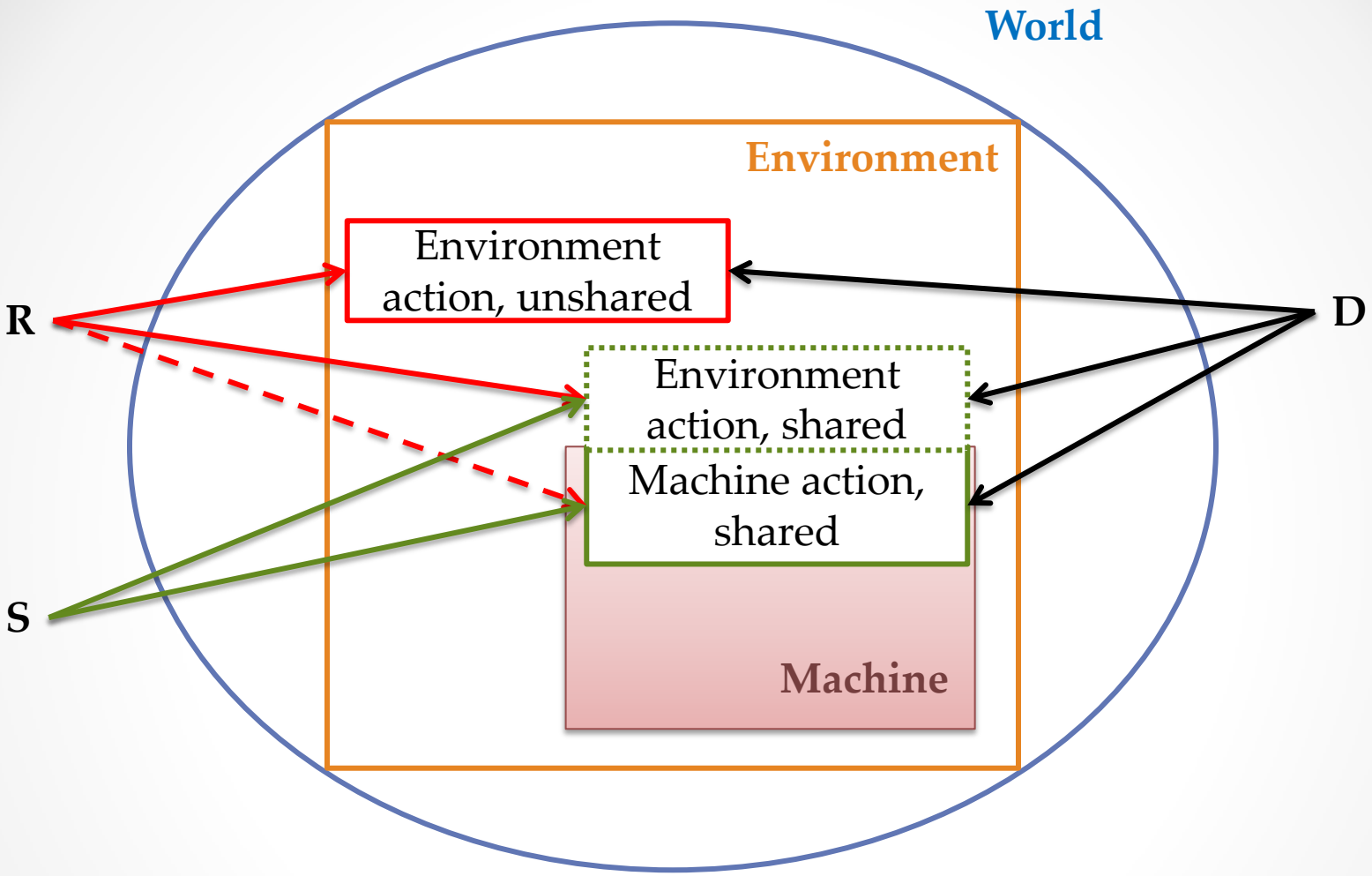
Zave&Jackson's theory

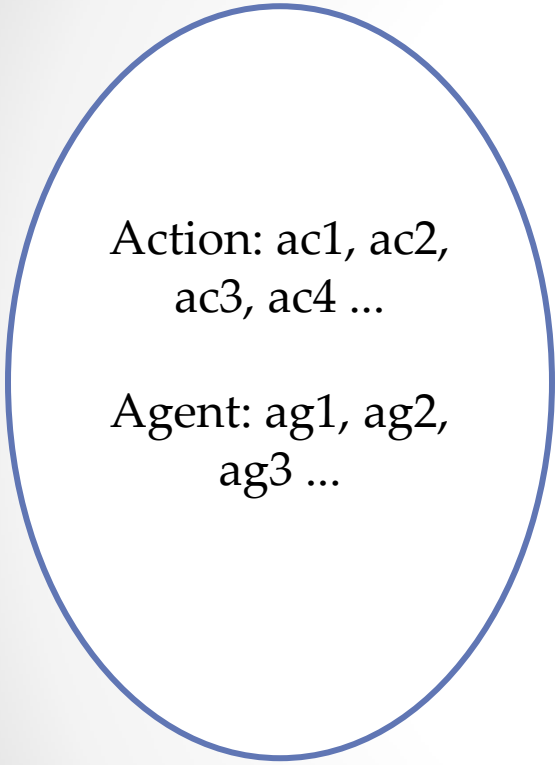
- $\boldsymbol{D}, \boldsymbol{S} \vdash \boldsymbol{R}$
- R indicates the Requirement, usually understood as early requirement
- S indicates the Specification, usually understood as late requirement refined from early requirement
- D indicates the Domain assumption, usually understood as the situations supporting the specification to fulfill the requirement

But these are not the original ideas of Zave and Jackson, terms are used ambiguously.

- $\boldsymbol{D}, \boldsymbol{S} \vdash \boldsymbol{R}$
- Property (*constrains on the T-BOX in DL*)
  - opative property
    - R: requirement
    - S: specification (implementable)
  - indicative property
    - D: domain assumption

The formula is based on the model theory, it could be represented as a knowledge base through a DL.


World
Environment
Environment action, unshared
Environment action, shared
Machine action, shared
Machine
R
S
D



Language={insertCoin(x), receiveCoin (x), buttonPress(x), coffeeOut(x), follow (x, y),…}

A-Box={
insertCoint(ac1), receiveCoin (ac2),
buttonPress(ac3), coffeeOut(ac4),
Agent(ag1), Agent(ag2),…}

T-Box={
R: $\forall x.\, insertCoin(x) \rightarrow \exists y.\, receiveCoin(y)$
S: $\forall x.\, buttonPress(x) \rightarrow \exists y.\, coffeeOut(y)$
D: $\neg\exists x.\, receiveCoin(x) \rightarrow \neg\exists ac2.\, coffeeOut(y)$
…
}

Possible Models
M1:
insertCoint={ac1},receiveCoin={ac2},
buttonPress ={ac3}, coffeeOut ={ac4},
follow={<ac1,ac2>,<ac3,ac4> …}

Comparasion

- After all, the requirement, specification and domain assumption are constraints on the T-Box (on the action instances), they constraint the possible models according to the domain.

- Our proposal

  Requirement: set of desired state of affairs

  Specification: set of functions

  Domain assumptions: part of the state of affairs which holds during the execution of protocol

Conclusion

- A new definition of computer software
  several concepts are discussed
- Ontologies of computer software/program
  the relation between concepts are discussed
- An comparison with Zave&Jackson's theory
  clarify the concepts and emphasize the differences
-

Future work

- Social Artefact and Information Object
- Mental state underlying the requirement
  - desire
  - intention
- Identity of the software
  - species level
  - instance level

The end

Thanks!

QUESTIONS?