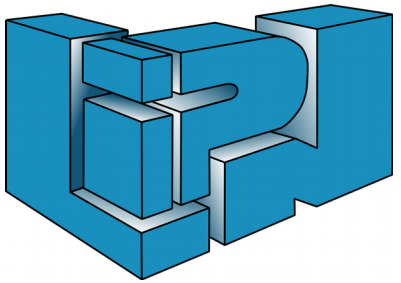
A new Model for Scalable θ-subsumption

Hippolyte Léger
Dominique Bouthinon
Mustapha Lebbah
Hanane Azzag

Laboratoire d'Informatique de Paris Nord - UMR CNRS 7030

Université Paris 13, Sorbonne Paris Cité, France



Outline

- Introduction
- Scalable θ-subsumption based on an actor model
- Experiments
- Conclusions and perspectives

Introduction

θ-subsumption

C θ-subsumes D iff $\exists\theta, C\theta \subseteq D$

θ-subsumption

C θ-subsumes D iff $\exists\theta$, $C\theta \subseteq D$

Example :

- C = h(U) ← p(U,X,Y), q(Y,Z), r(Z,Z,T)
- D = h(a) ← p(a,b,c), q(c,e), r(e,a,c), q(d,f), r(a,a,d), r(e,e,g)

θ-subsumption

C θ-subsumes D iff $\exists\theta$, $C\theta \subseteq D$

Example :

- C = h(U) ← p(U,X,Y), q(Y,Z), r(Z,Z,T)
- D = h(a) ← **p(a,b,c)**, **q(c,e)**, r(e,a,c), q(d,f), r(a,a,d), **r(e,e,g)**

$C\theta \subseteq D$, with θ={U/a,X/b,Y/c,Z/e,T/g}

θ-subsumption

C θ-subsumes D iff $\exists\theta$, $C\theta \subseteq D$

Example :

- C = h(U) ← p(U,X,Y), q(Y,Z), r(Z,Z,T)
- D = h(a) ← **p(a,b,c)**, **q(c,e)**, r(e,a,c), q(d,f), r(a,a,d), **r(e,e,g)**

$C\theta \subseteq D$, with θ={U/a,X/b,Y/c,Z/e,T/g}

The θ-subsumption test is the coverage relation between hypotheses and examples in most ILP systems

θ-subsumption

C θ-subsumes D iff $\exists\theta$, $C\theta \subseteq D$

Example :

- C = h(U) ← p(U,X,Y), q(Y,Z), r(Z,Z,T)
- D = h(a) ← **p(a,b,c)**, **q(c,e)**, r(e,a,c), q(d,f), r(a,a,d), **r(e,e,g)**

$C\theta \subseteq D$, with θ={U/a,X/b,Y/c,Z/e,T/g}

The θ-subsumption test is the coverage relation between hypotheses and examples in most ILP systems

Worst case time complexity : $O(|D|^{|C|})$

State-of-the-art θ-subsumption algorithms

- Django (Maloberti et al. 2004)
- Resumer (Kuzelka et al. 2008)
- Subsumer (Santos et al. 2010)
- ...

State-of-the-art θ-subsumption algorithms

- Django (Maloberti et al. 2004)
- Resumer (Kuzelka et al. 2008)
- Subsumer (Santos et al. 2010)
- ...

Efficient, but **not scalable**

problems with huge clauses

Our goal : scalable θ-subsumption

Challenge :

Build a θ-subsumption engine that can be **parallelized and distributed** on large clusters

Our goal : scalable θ-subsumption

Challenge :

Build a θ-subsumption engine that can be **parallelized and distributed** on large clusters

Scalability :

- Use **new paradigms** (MapReduce, Spark, Actor Model, ...)
- Design **a new model** of θ-subsumption

Scalable θ-subsumption based on an actor model

Classic θ-subsumption process

Example :

C = h(U) ← p(U,X,Y), q(Y,Z), r(Z,Z,T)

D = h(a) ← p(a,b,c), q(c,e), r(e,a,c), q(d,f), r(a,a,d), r(e,e,g)

Classic θ-subsumption process

Example :

C = h(U) ← p(U,X,Y), q(Y,Z), r(Z,Z,T)

D = h(a) ← p(a,b,c), q(c,e), r(e,a,c), q(d,f), r(a,a,d), r(e,e,g)

1) Unify the head of clauses :

➔ α = {U/a} ⇒ *head(C)*α = *head(D)*

Classic θ-subsumption process

Example :

C = h(U) ← p(U,X,Y), q(Y,Z), r(Z,Z,T)

D = h(a) ← p(a,b,c), q(c,e), r(e,a,c), q(d,f), r(a,a,d), r(e,e,g)

1) Unify the head of clauses :

➔ α = {U/a} ⇒ *head(C)*α = *head(D)*

2) Check that *body(C)α* θ-subsumes *body(D)*

➔ comes down to checking the θ-subsumption of two conjonctions

Classic θ-subsumption process

Example :

C = h(U) ← p(U,X,Y), q(Y,Z), r(Z,Z,T)

D = h(a) ← p(a,b,c), q(c,e), r(e,a,c), q(d,f), r(a,a,d), r(e,e,g)

1) Unify the head of clauses :

➔ α = {U/a} ⇒ *head(C)*α = *head(D)*

2) Check that *body(C)α* θ-subsumes *body(D)*

➔ comes down to checking the θ-subsumption of two conjonctions

A = {p(a,X,Y), q(Y,Z), r(Z,Z,T)}

B = {p(a,b,c), q(c,e), r(e,a,c), q(d,f), r(a,a,d), r(e,e,g)}

Split θ-subsumption

Let A = $\{a_1, ..., a_n\}$ be a conjunction of literals and B be a conjunction of ground literals.

<u>Split the θ-subsumption process :</u>

A θ-subsumes B ⇔ there exists a set of **compatible substitutions** $\theta_1, \theta_2, ..., \theta_n$ such that $a_1\theta_1 \in B$, $a_2\theta_2 \in B$, ..., $a_n\theta_n \in B$

Split θ-subsumption

Let A = $\{a_1, ..., a_n\}$ be a conjunction of literals and B be a conjunction of ground literals.

Split the θ-subsumption process :

A θ-subsumes B ⇔ there exists a set of **compatible substitutions** $\theta_1, \theta_2, ..., \theta_n$ such that $a_1\theta_1 \in B$, $a_2\theta_2 \in B$, ..., $a_n\theta_n \in B$

Back to our example :

A = {p(a,X,Y), q(Y,Z), r(Z,Z,T)}

B = {p(a,b,c), q(c,e), r(e,a,c), q(d,f), r(a,a,d), r(e,e,g)}

Split θ-subsumption

Let A = $\{a_1, ..., a_n\}$ be a conjunction of literals and B be a conjunction of ground literals.

Split the θ-subsumption process :

A θ-subsumes B ⇔ there exists a set of **compatible substitutions** $\theta_1, \theta_2, ..., \theta_n$ such that $a_1\theta_1 \in B$, $a_2\theta_2 \in B$, ..., $a_n\theta_n \in B$

Back to our example :

A = {p(a,X,Y), q(Y,Z), r(Z,Z,T)}

B = {p(a,b,c), q(c,e), r(e,a,c),
q(d,f), r(a,a,d),
r(e,e,g)}

Split θ-subsumption

Let A = {a_1, ..., a_n} be a conjunction of literals and B be a conjunction of ground literals.

Split the θ-subsumption process :

A θ-subsumes B ⇔ there exists a set of **compatible substitutions** θ_1, θ_2, ..., θ_n such that $a_1\theta_1 \in B$, $a_2\theta_2 \in B$, ..., $a_n\theta_n \in B$

Back to our example :

A = {p(a,X,Y), q(Y,Z), r(Z,Z,T)}

B = {**p(a,b,c)**, **q(c,e)**, r(e,a,c), q(d,f), r(a,a,d), **r(e,e,g)**}

→ θ_1 = {X/b,Y/c}

→ θ_2 = {Y/c,Z/e}

→ θ_3 = {Z/e,T/g}

Split θ-subsumption

Let A = {a_1, ..., a_n} be a conjunction of literals and B be a conjunction of ground literals.

Split the θ-subsumption process :

A θ-subsumes B ⇔ there exists a set of **compatible substitutions** θ_1, θ_2, ..., θ_n such that $a_1\theta_1 \in B$, $a_2\theta_2 \in B$, ..., $a_n\theta_n \in B$

Back to our example :

A = {p(a,X,Y), q(Y,Z), r(Z,Z,T)}

B = {**p(a,b,c)**, **q(c,e)**, r(e,a,c), q(d,f), r(a,a,d), **r(e,e,g)**}

θ_1 = {X/b,Y/c}

θ_2 = {Y/c,Z/e}

θ_3 = {Z/e,T/g}

Model a parallel/distributed split θ-subsumption

Let A = {a_1, ..., a_n} be a conjunction of literals and B be a conjunction of ground literals.

Goal :

find a set of **compatible subsitutions** θ_1, θ_2, ..., θ_n such that $a_1\theta_1 \in B$, $a_2\theta_2 \in B$,..., $a_n\theta_n \in B$

Model a parallel/distributed split θ-subsumption

Let A = {a_1, ..., a_n} be a conjunction of literals and B be a conjunction of ground literals.

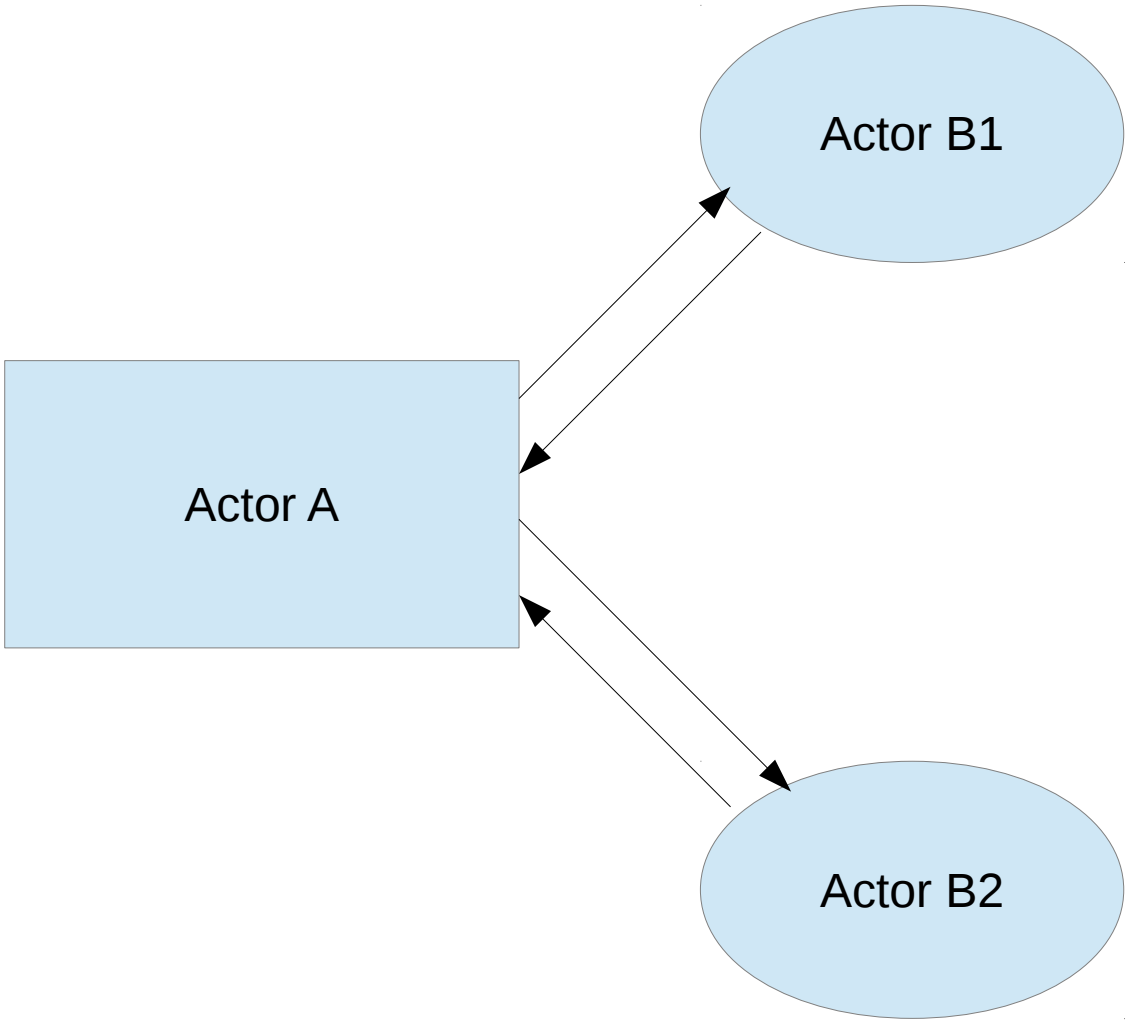
Goal :

find a set of **compatible subsitutions** θ_1, θ_2, ..., θ_n such that $a_1\theta_1 \in B$, $a_2\theta_2 \in B$,..., $a_n\theta_n \in B$

Approach :

Use a model based on actors

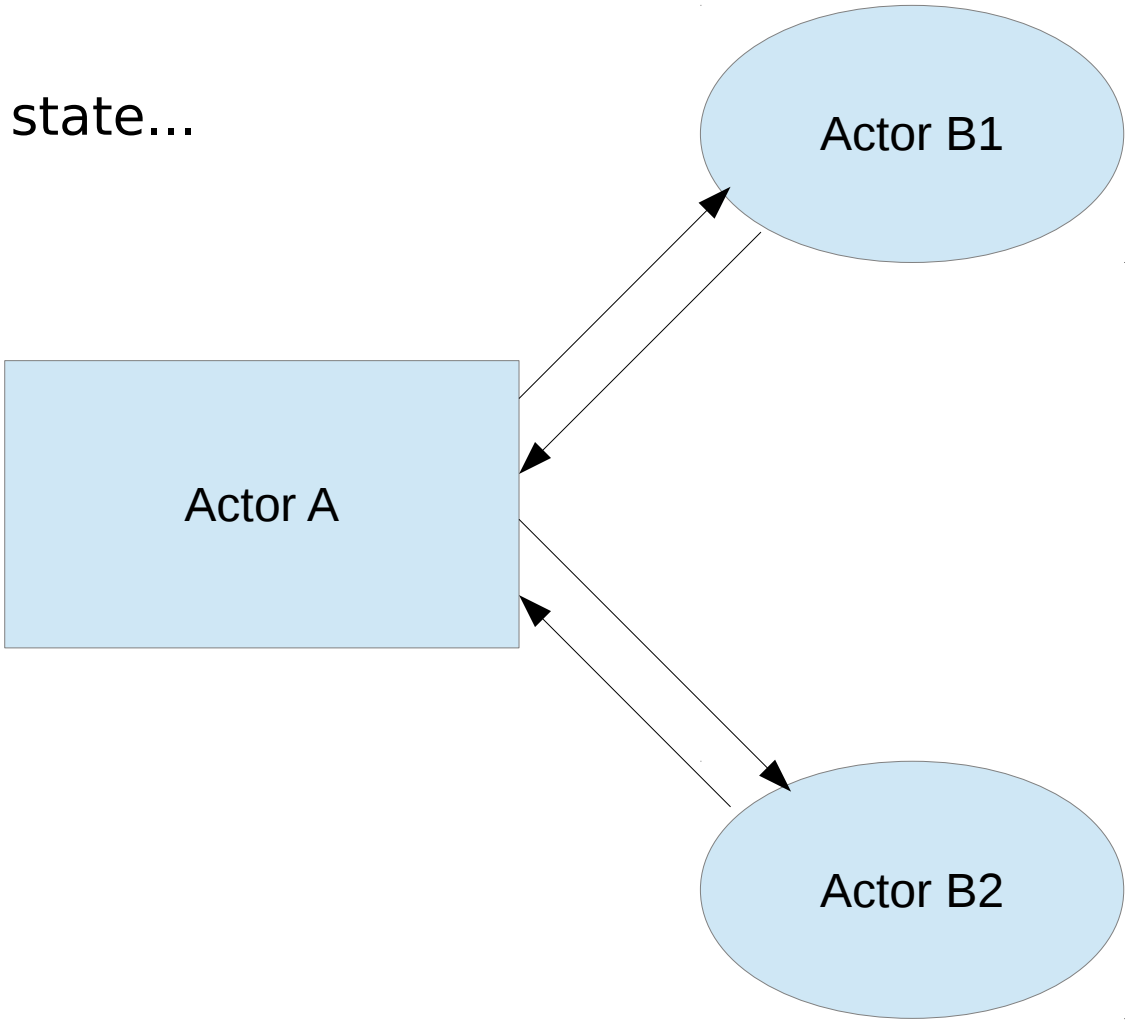
Actor network example



Actor network example

Actors can :

- Communicate...
- Have an internal state...
- Perform tasks...

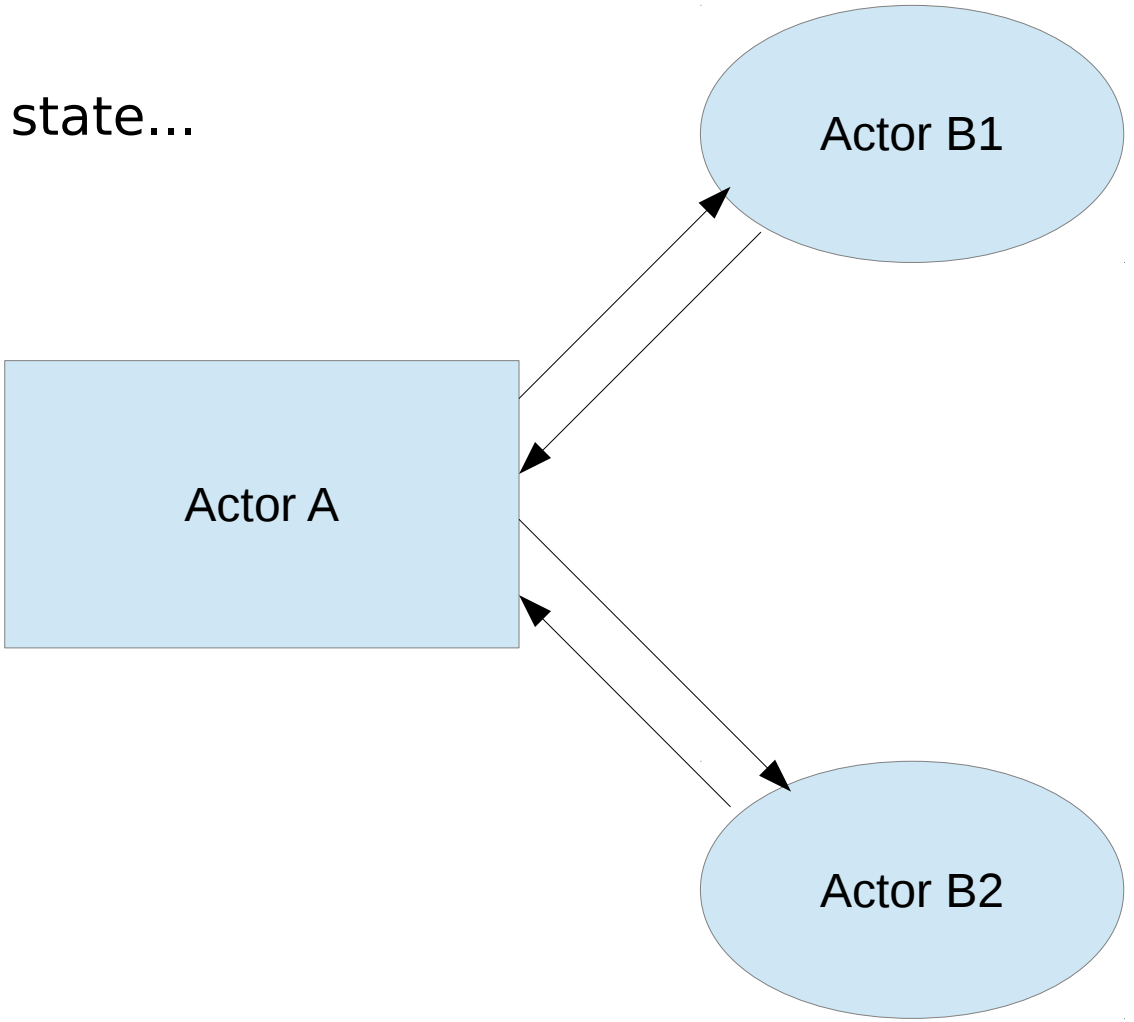


Actor network example

Actors can :

- Communicate...
- Have an internal state...
- Perform tasks...

... Asynchronously !



Split θ-subsumption based on an actor Model

A = {p(a,X,Y), q(Y,Z), r(Z,Z,T)}

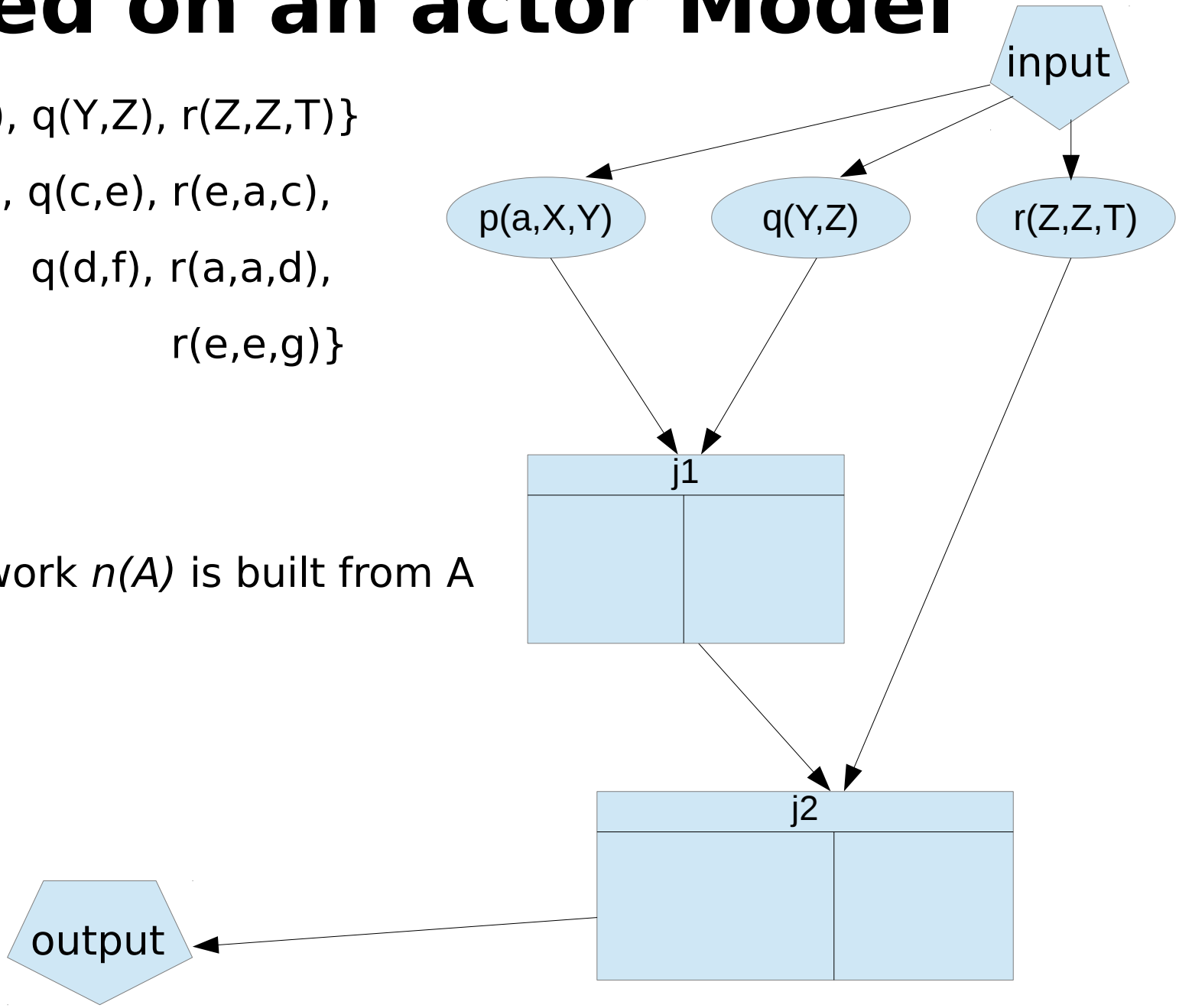
B = {p(a,b,c), q(c,e), r(e,a,c),
q(d,f), r(a,a,d),
r(e,e,g)}

Split θ-subsumption based on an actor Model

A = {p(a,X,Y), q(Y,Z), r(Z,Z,T)}

B = {p(a,b,c), q(c,e), r(e,a,c),
q(d,f), r(a,a,d),
r(e,e,g)}

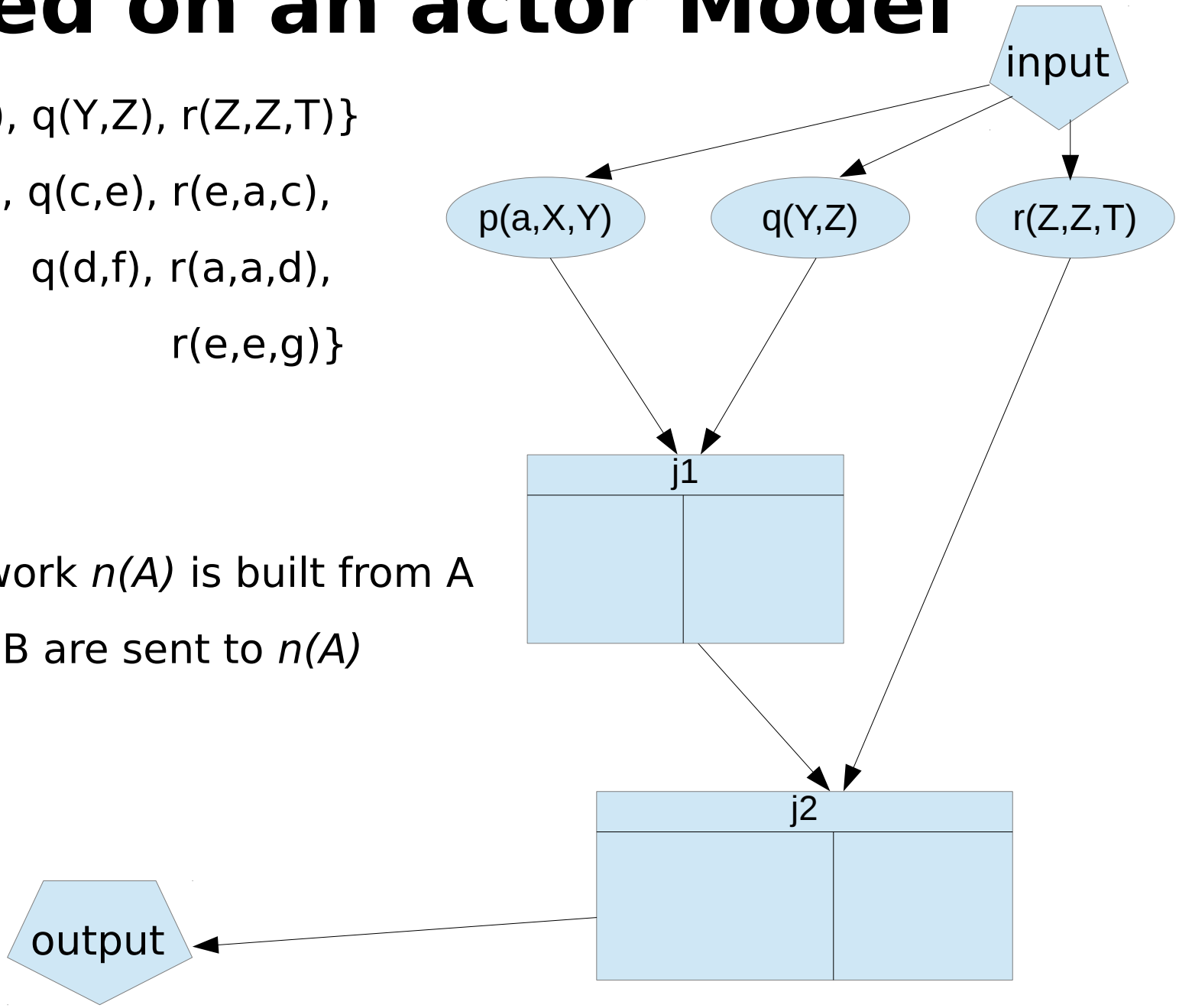
1) an actor network *n(A)* is built from A



Split θ-subsumption based on an actor Model

A = {p(a,X,Y), q(Y,Z), r(Z,Z,T)}

B = {p(a,b,c), q(c,e), r(e,a,c),
q(d,f), r(a,a,d),
r(e,e,g)}

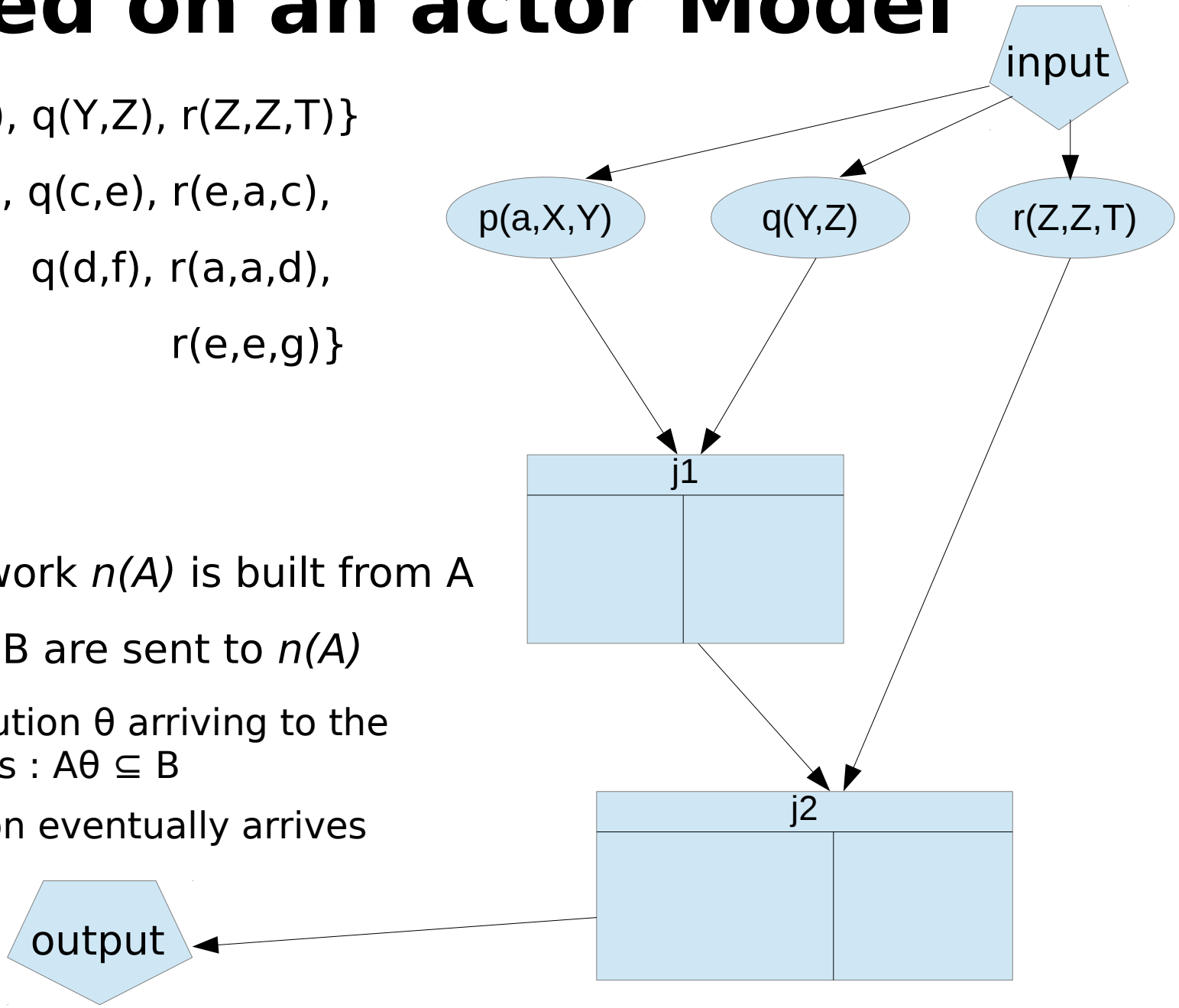


1) an actor network *n(A)* is built from A
2) the atoms of B are sent to *n(A)*

Split θ-subsumption based on an actor Model

A = {p(a,X,Y), q(Y,Z), r(Z,Z,T)}

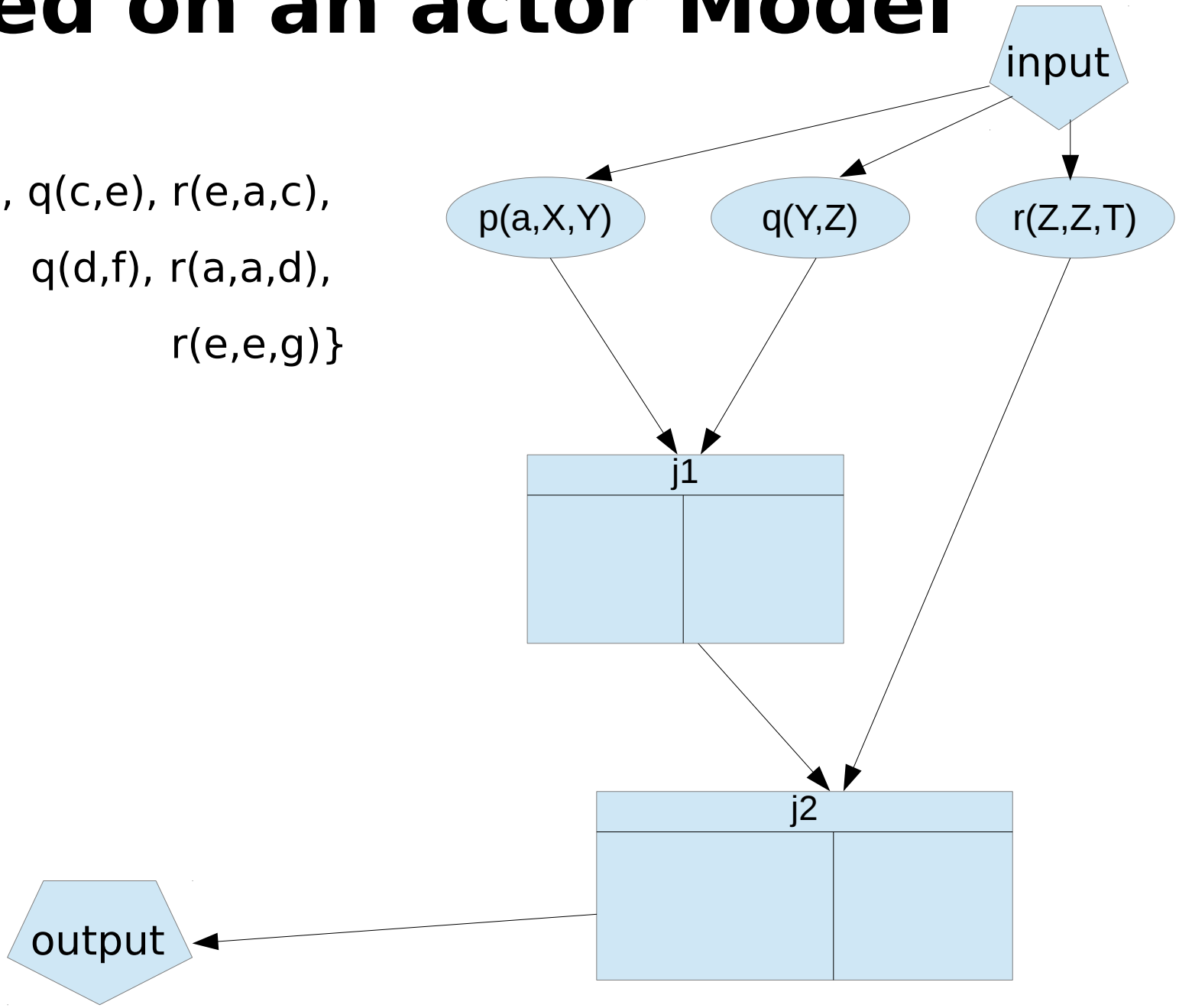
B = {p(a,b,c), q(c,e), r(e,a,c),
q(d,f), r(a,a,d),
r(e,e,g)}



1) an actor network *n(A)* is built from A
2) the atoms of B are sent to *n(A)*
   - each substitution θ arriving to the ouput verifies : Aθ ⊆ B
   - Every solution eventually arrives

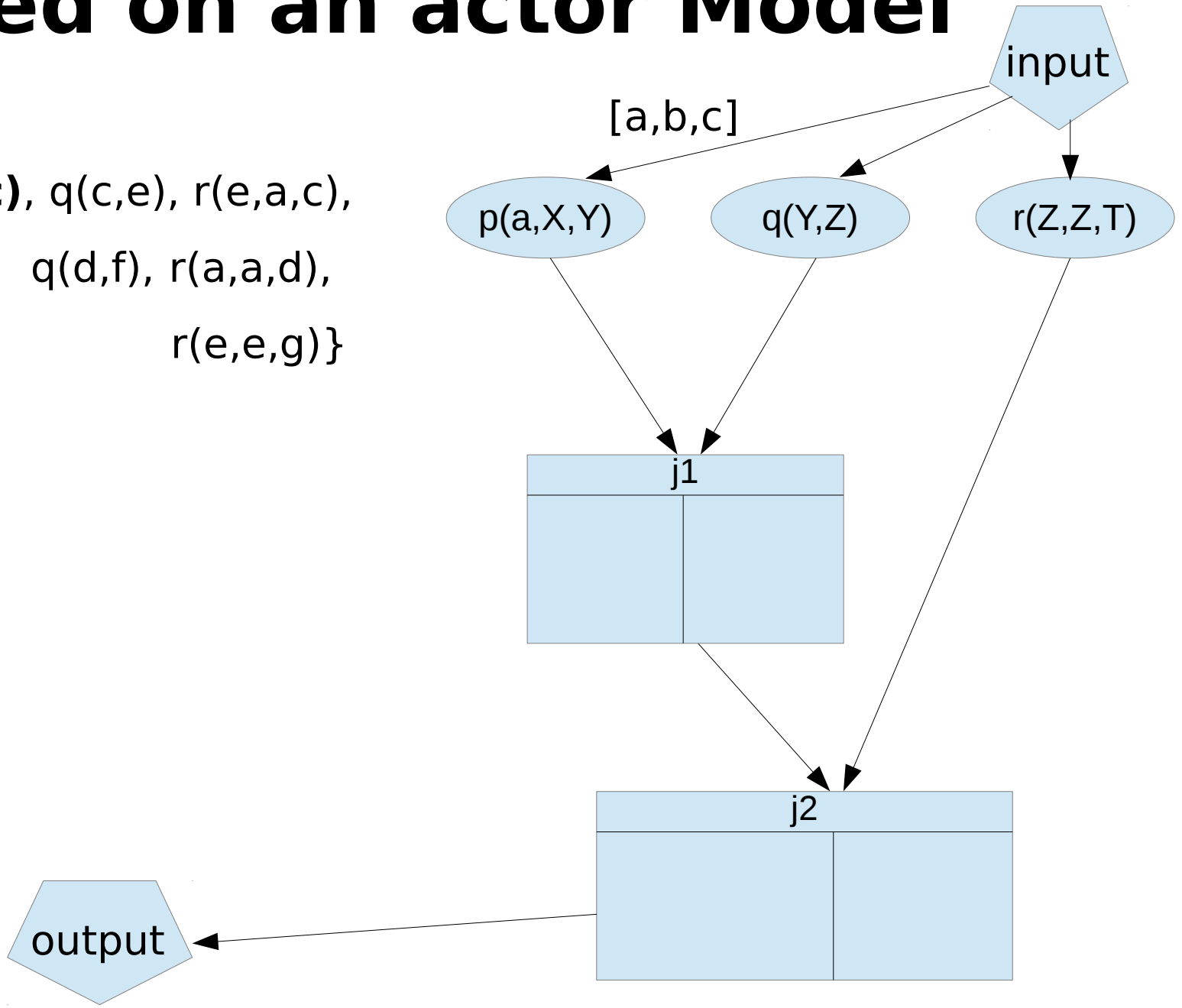
Split θ-subsumption based on an actor Model

B = {p(a,b,c), q(c,e), r(e,a,c),
q(d,f), r(a,a,d),
r(e,e,g)}



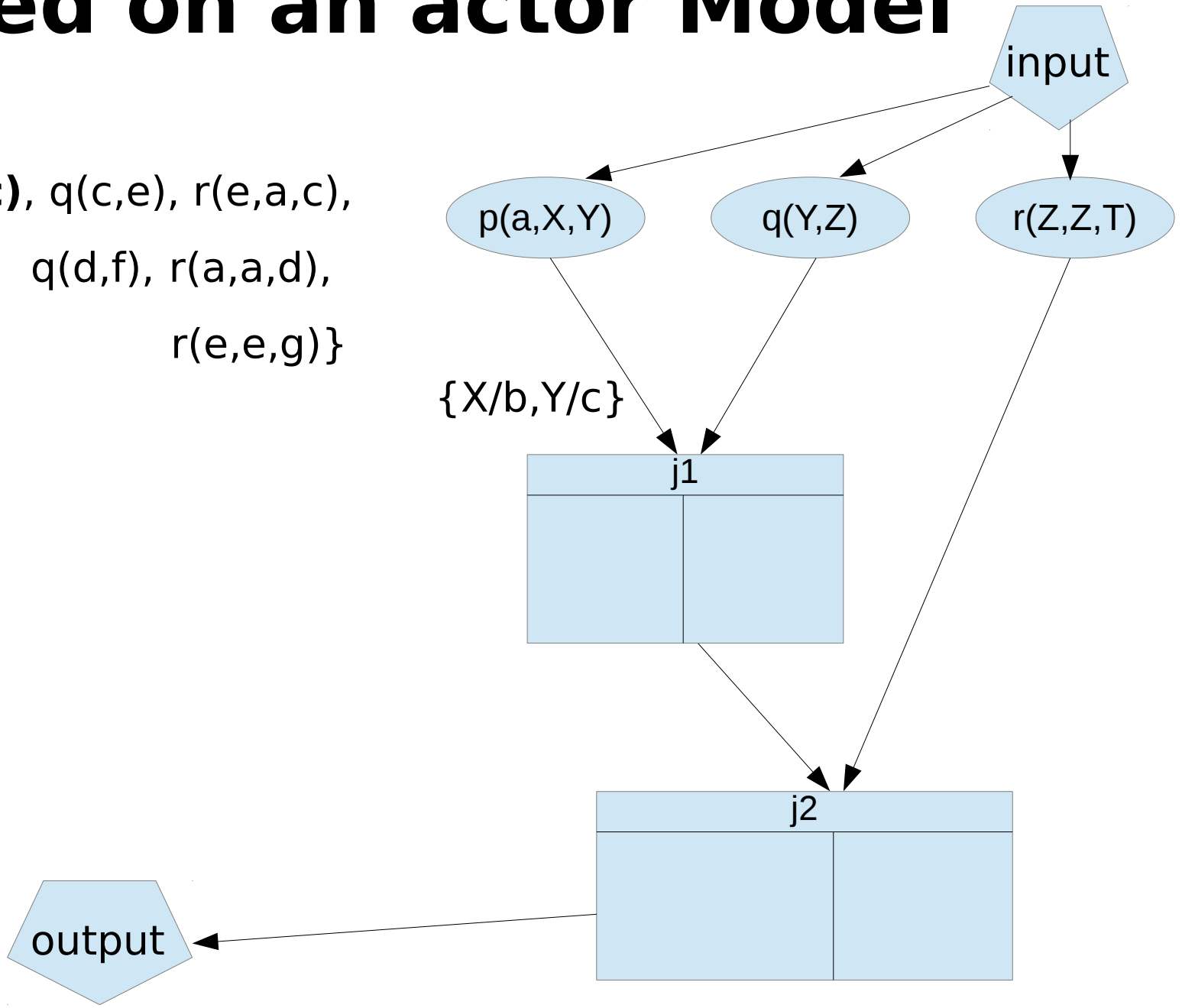
Split θ-subsumption based on an actor Model

B = {**p(a,b,c)**, q(c,e), r(e,a,c),
q(d,f), r(a,a,d),
r(e,e,g)}



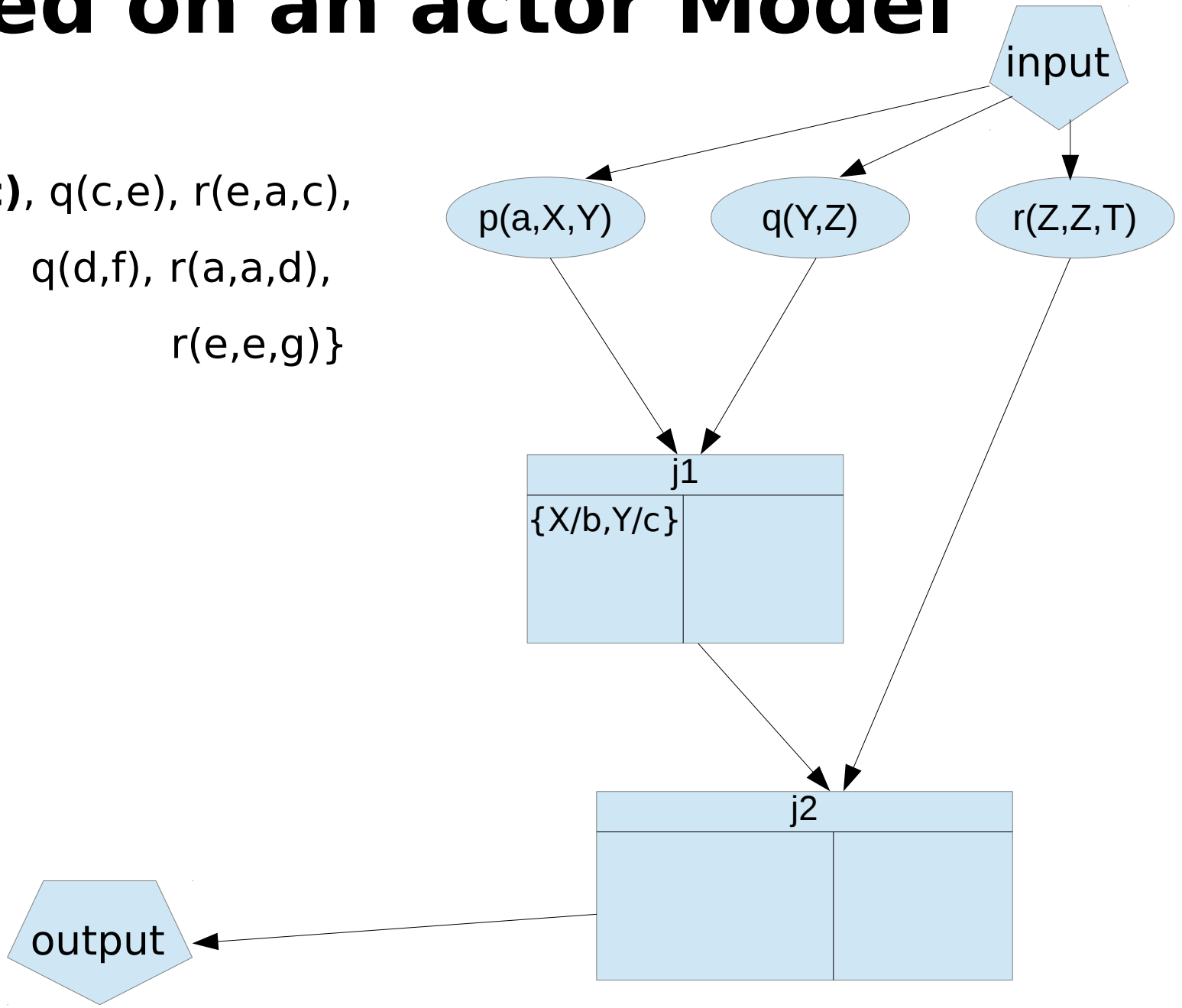
Split θ-subsumption based on an actor Model

B = {**p(a,b,c)**, q(c,e), r(e,a,c),
q(d,f), r(a,a,d),
r(e,e,g)}



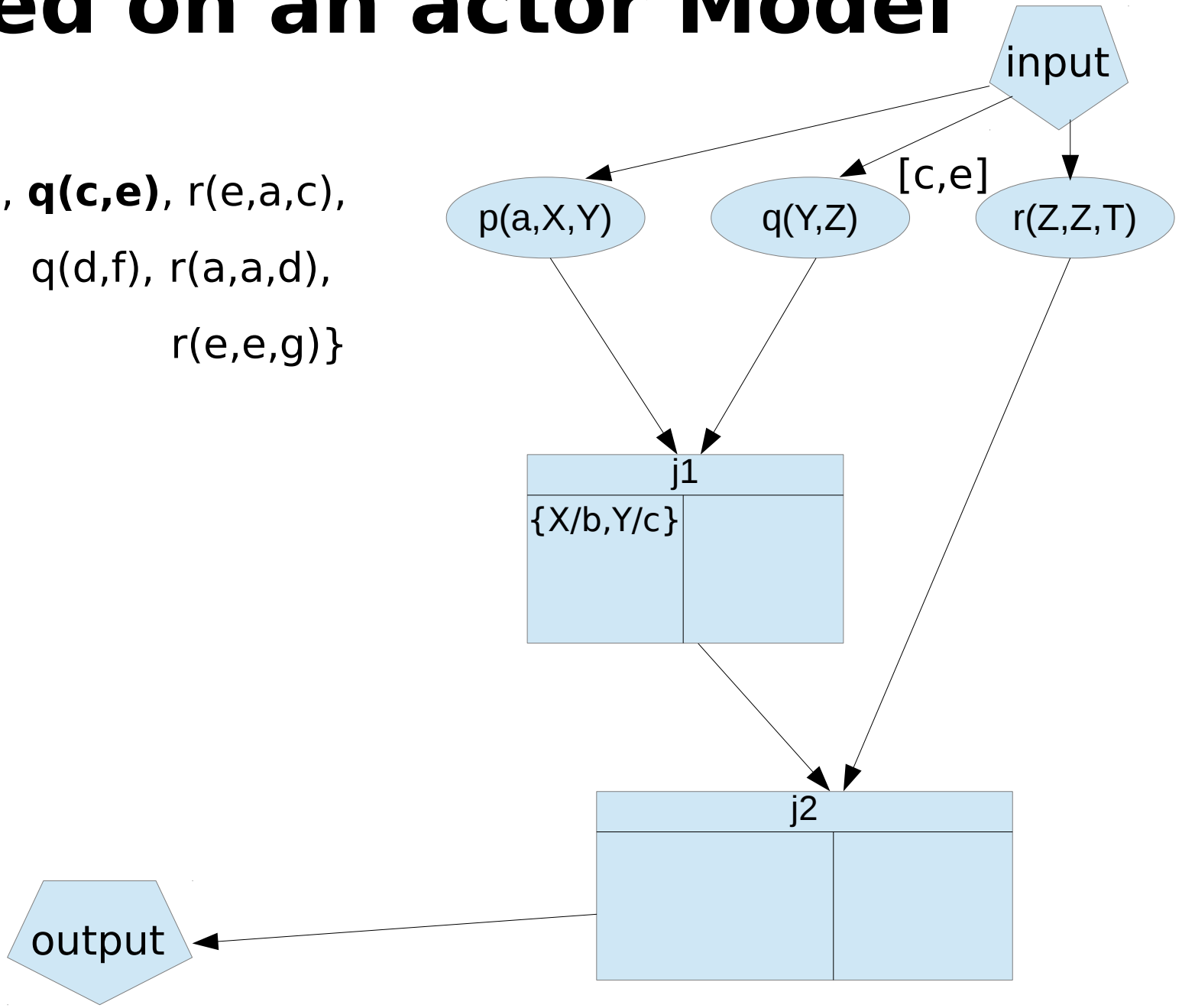
Split θ-subsumption based on an actor Model

B = {**p(a,b,c)**, q(c,e), r(e,a,c),
q(d,f), r(a,a,d),
r(e,e,g)}



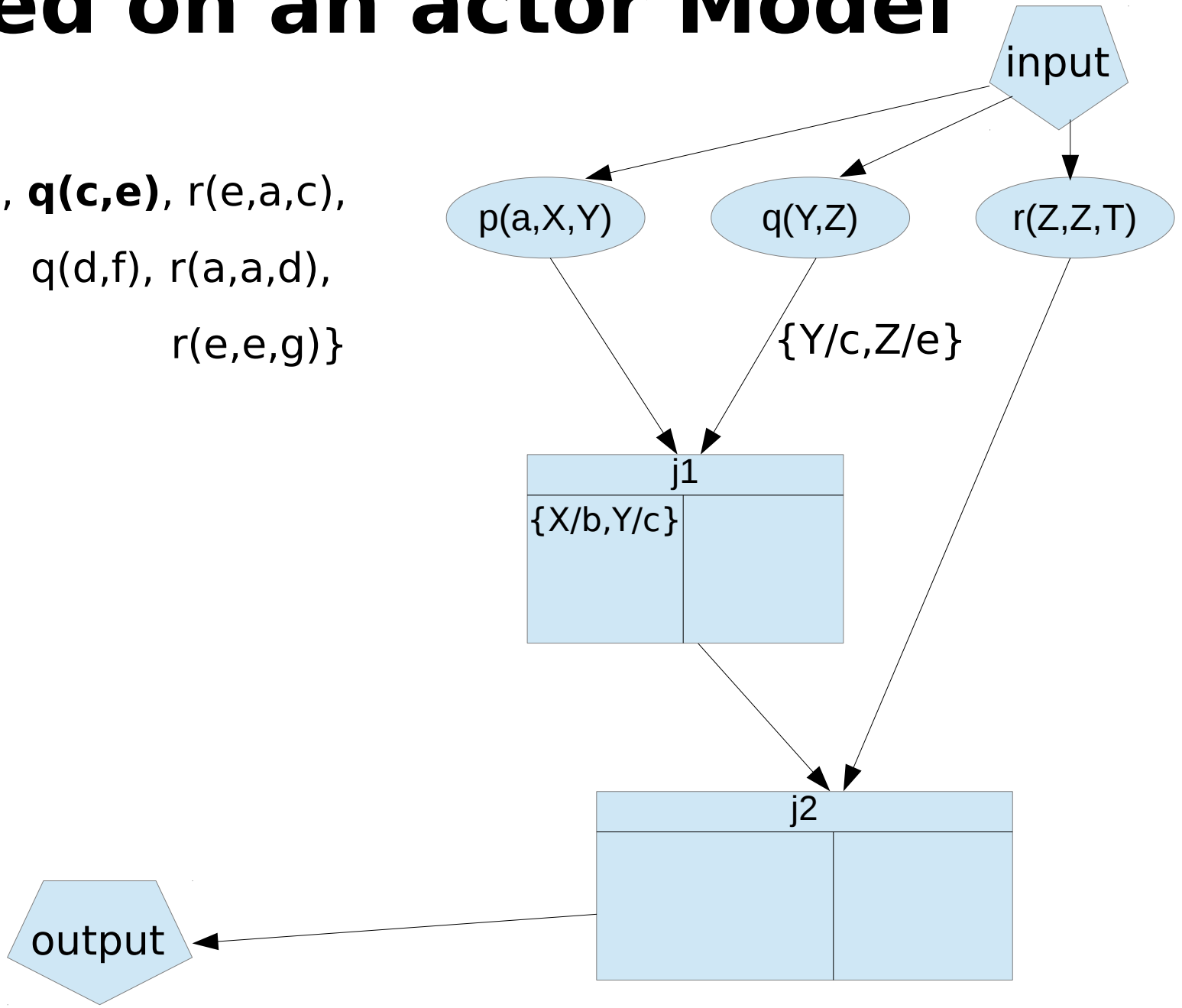
Split θ-subsumption based on an actor Model

B = {p(a,b,c), **q(c,e)**, r(e,a,c),
q(d,f), r(a,a,d),
r(e,e,g)}



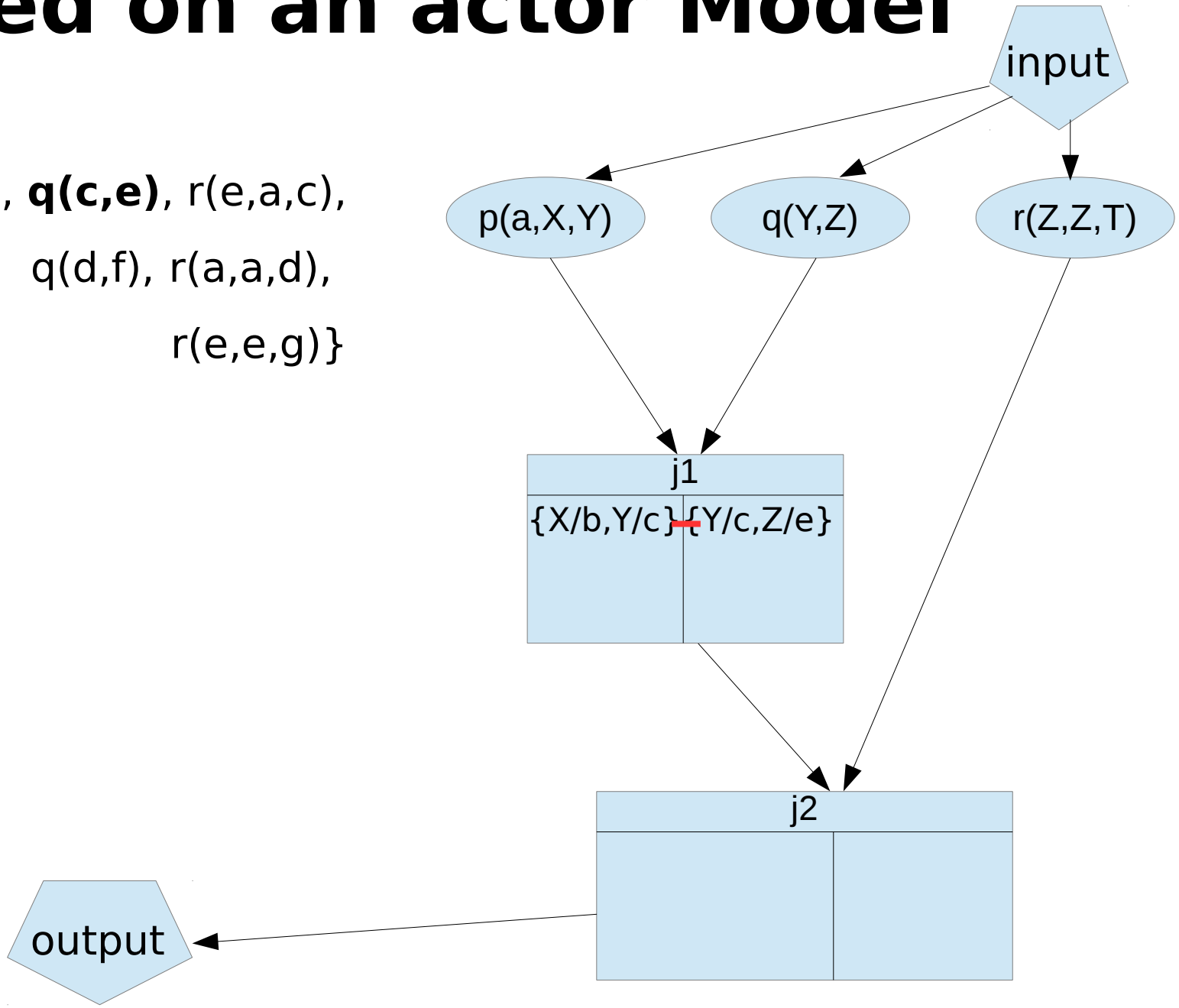
Split θ-subsumption based on an actor Model

B = {p(a,b,c), **q(c,e)**, r(e,a,c),
q(d,f), r(a,a,d),
r(e,e,g)}



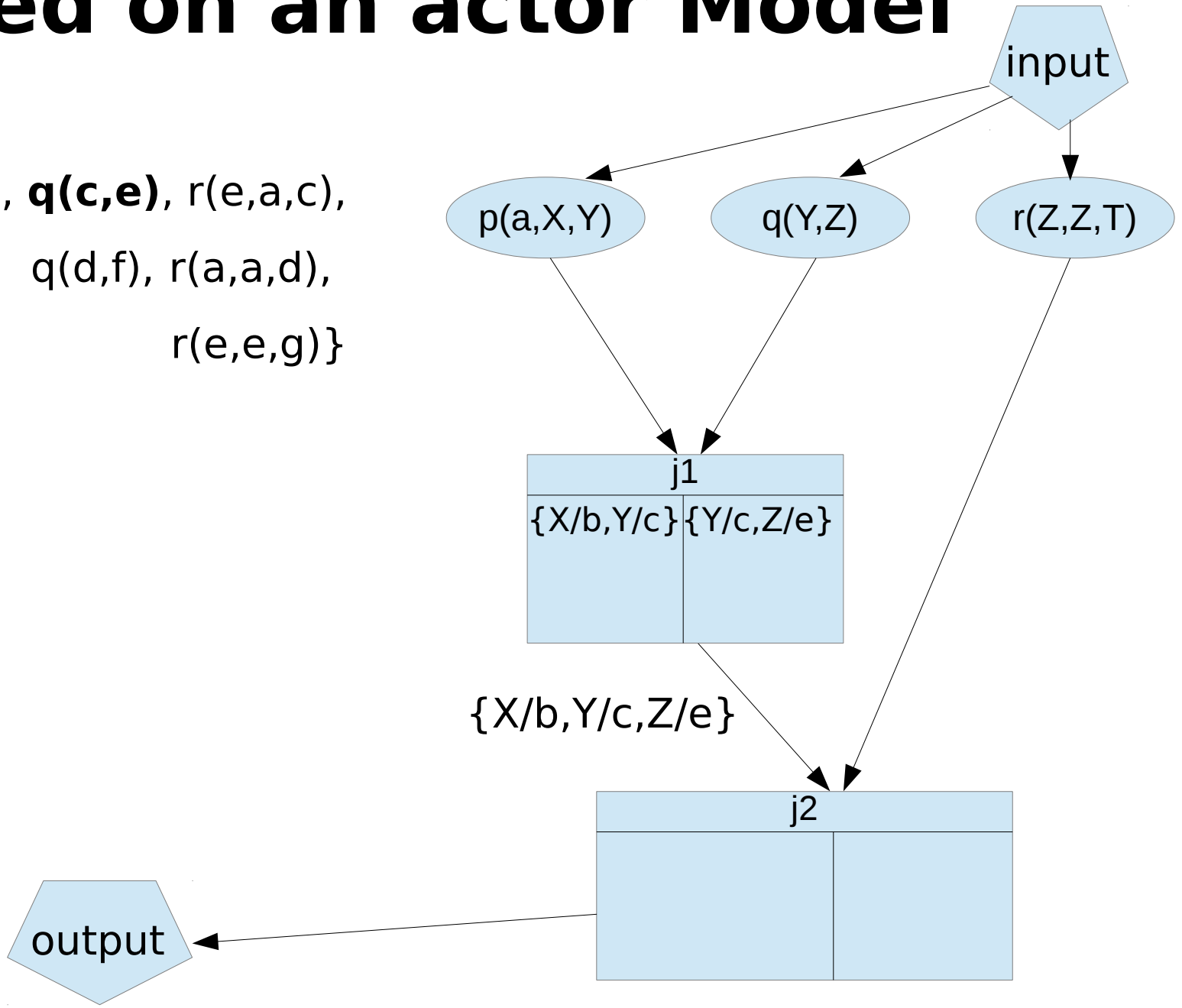
Split θ-subsumption based on an actor Model

B = {p(a,b,c), **q(c,e)**, r(e,a,c),
q(d,f), r(a,a,d),
r(e,e,g)}



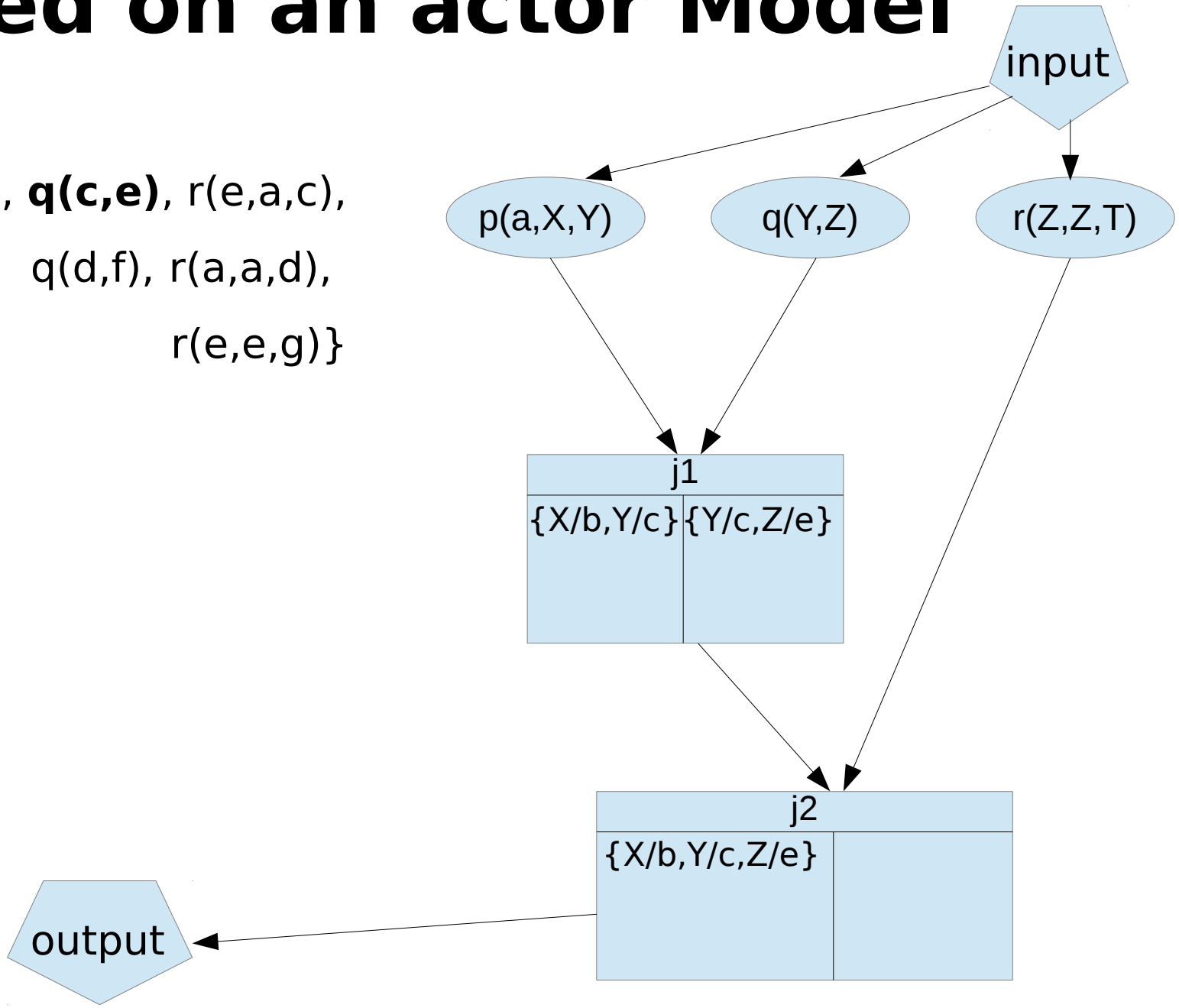
Split θ-subsumption based on an actor Model

B = {p(a,b,c), **q(c,e)**, r(e,a,c),
q(d,f), r(a,a,d),
r(e,e,g)}



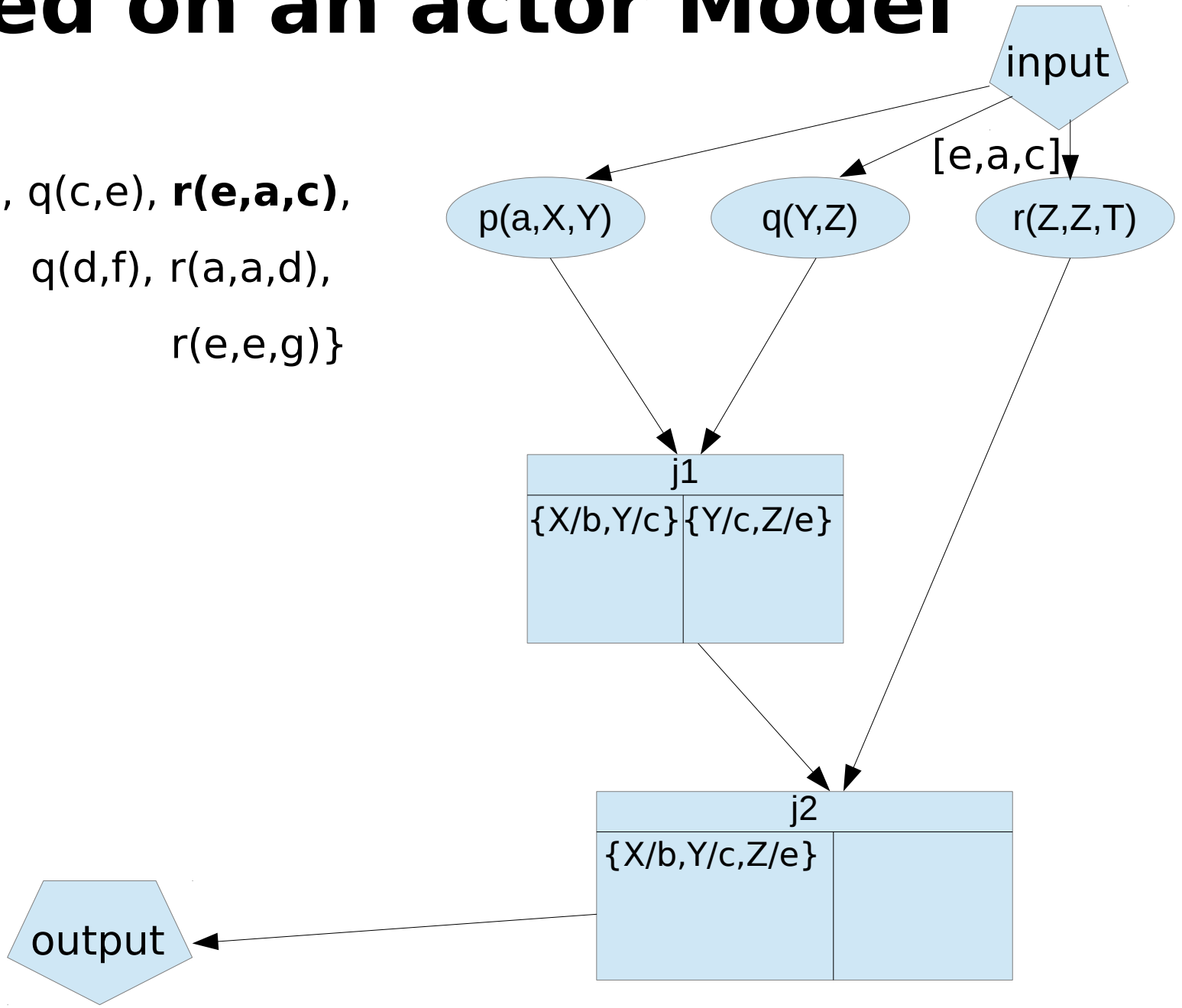
Split θ-subsumption based on an actor Model

B = {p(a,b,c), **q(c,e)**, r(e,a,c),
q(d,f), r(a,a,d),
r(e,e,g)}



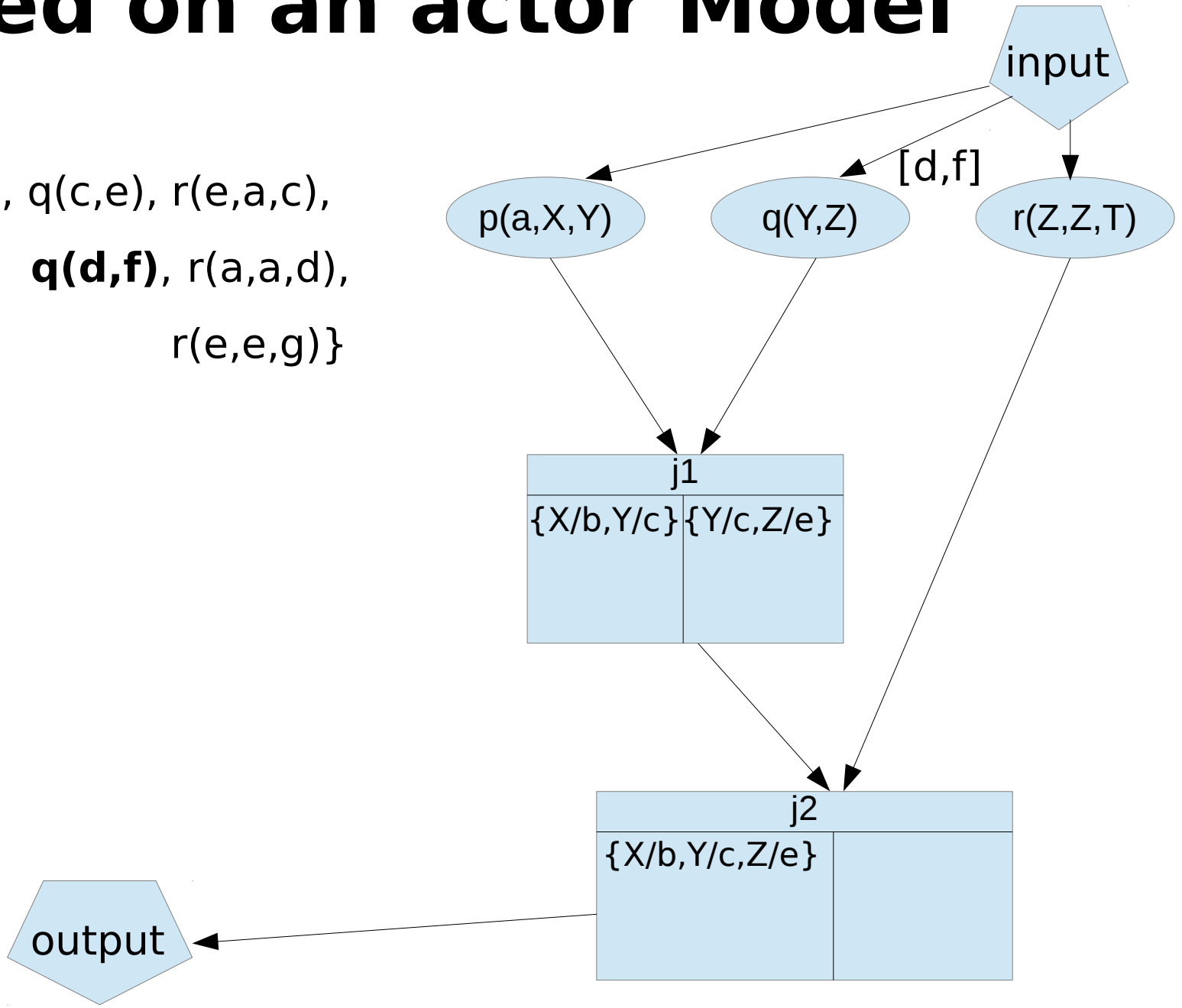
Split θ-subsumption based on an actor Model

B = {p(a,b,c), q(c,e), **r(e,a,c)**,
q(d,f), r(a,a,d),
r(e,e,g)}



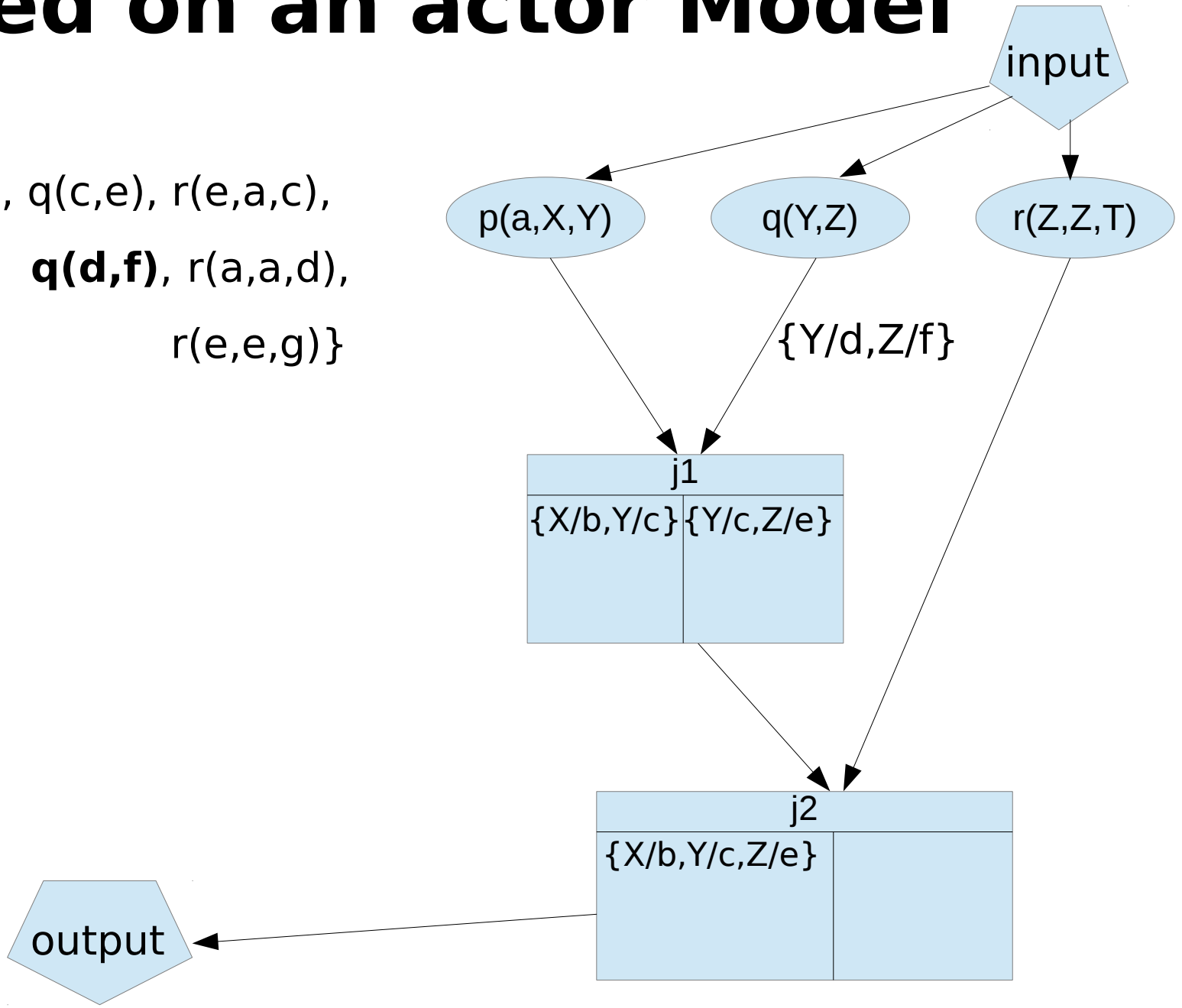
Split θ-subsumption based on an actor Model

B = {p(a,b,c), q(c,e), r(e,a,c),
q(d,f), r(a,a,d),
r(e,e,g)}



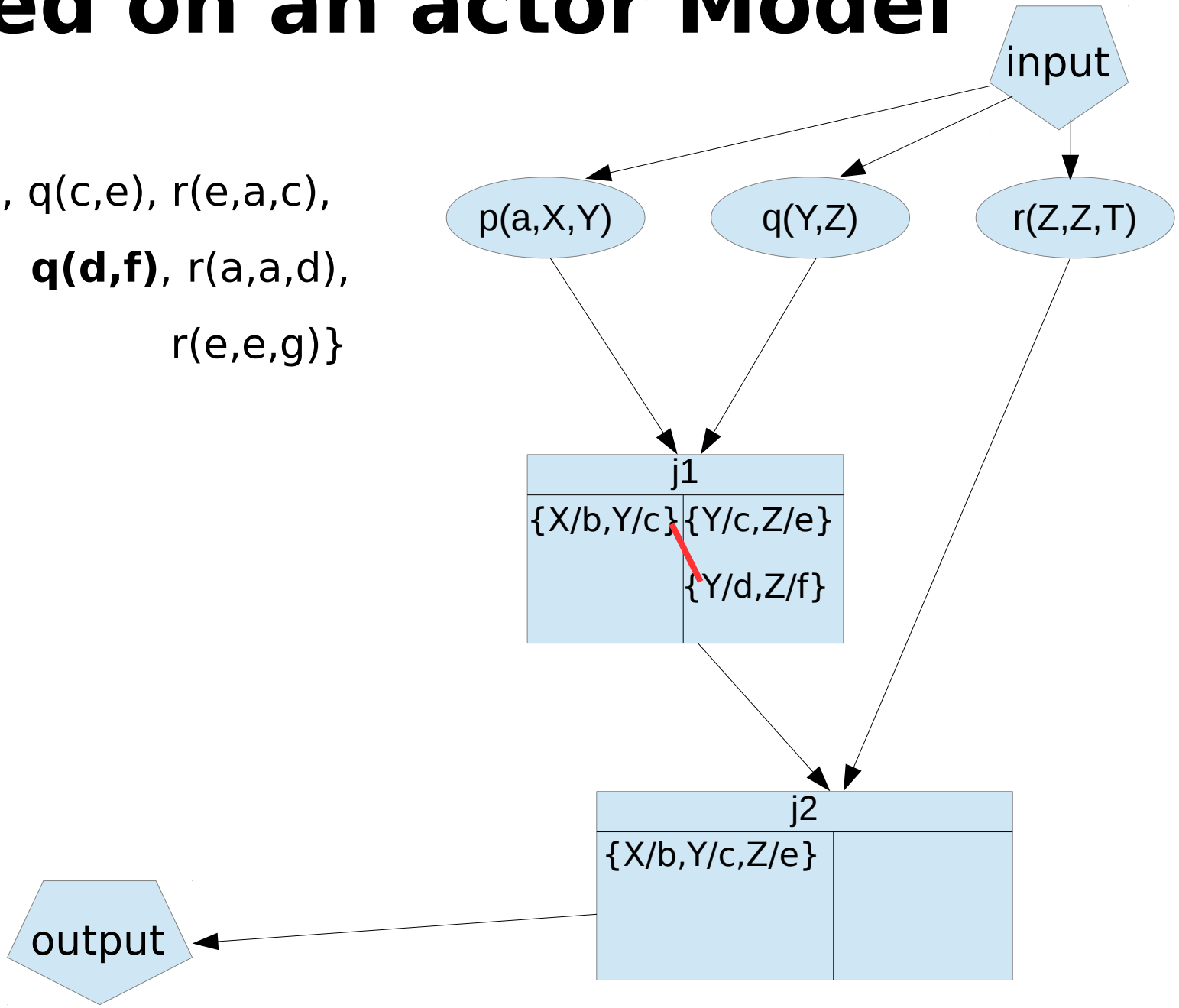
Split θ-subsumption based on an actor Model

B = {p(a,b,c), q(c,e), r(e,a,c),
q(d,f), r(a,a,d),
r(e,e,g)}



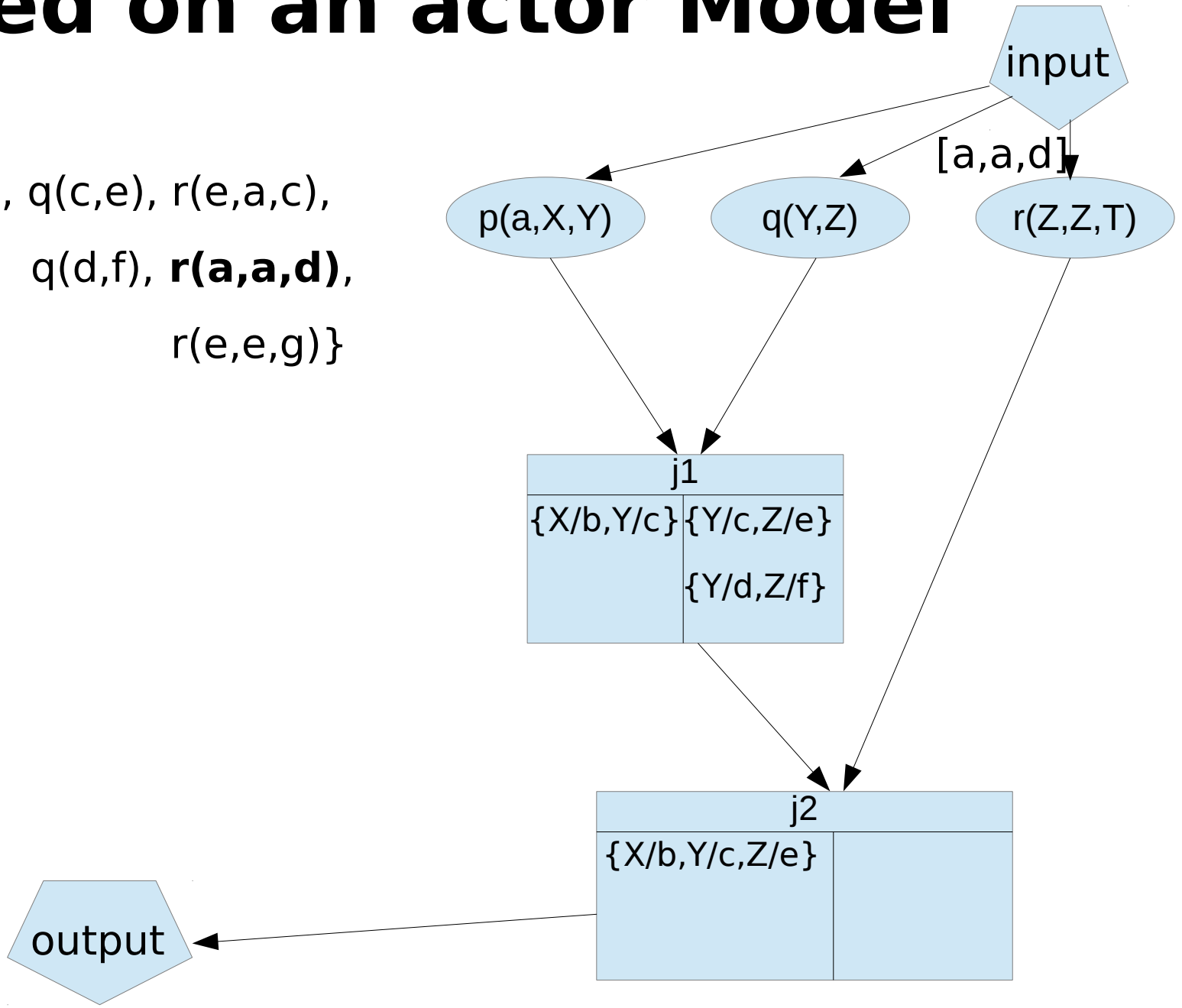
Split θ-subsumption based on an actor Model

B = {p(a,b,c), q(c,e), r(e,a,c),
q(d,f), r(a,a,d),
r(e,e,g)}



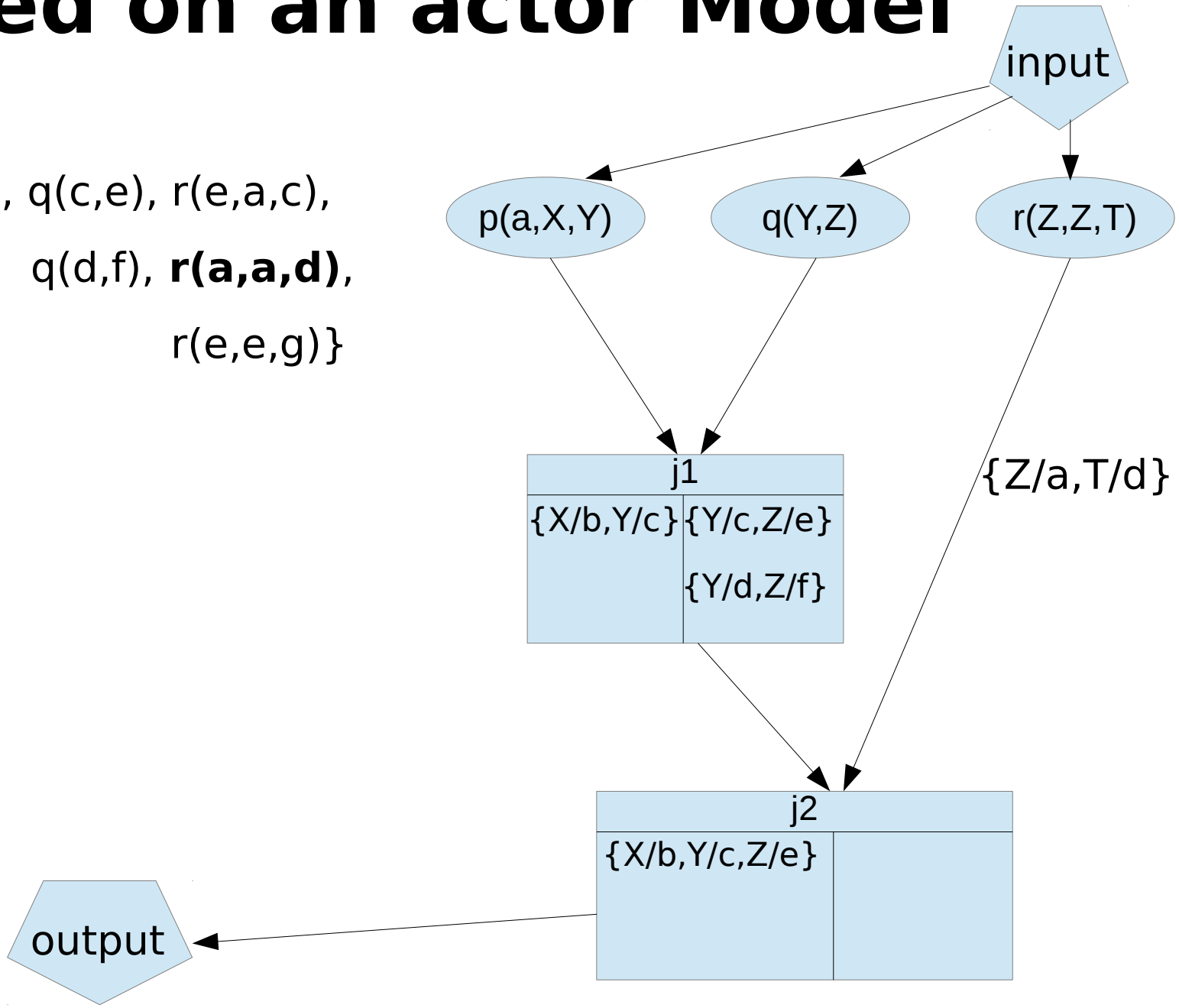
Split θ-subsumption based on an actor Model

B = {p(a,b,c), q(c,e), r(e,a,c),
q(d,f), **r(a,a,d)**,
r(e,e,g)}



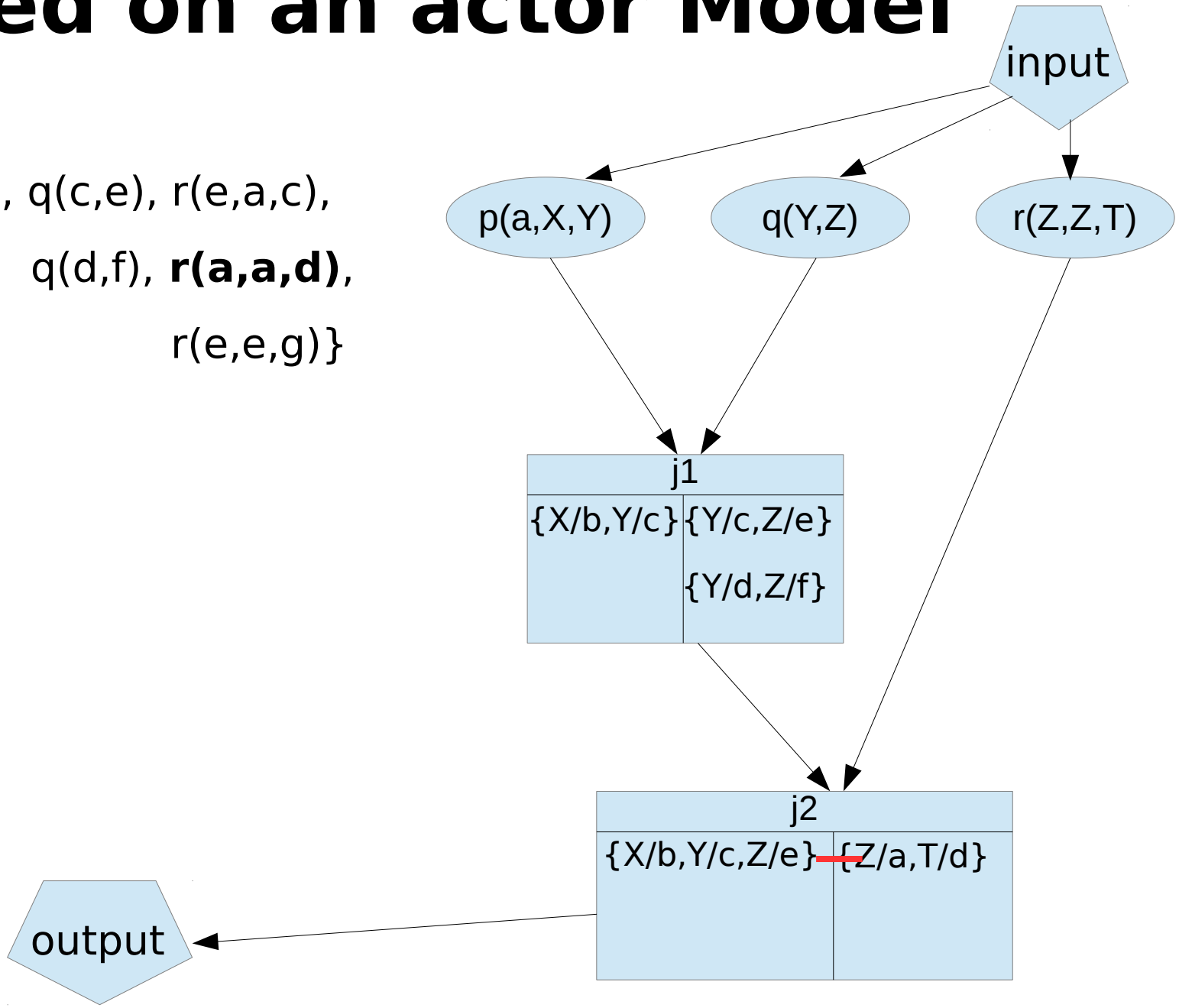
Split θ-subsumption based on an actor Model

B = {p(a,b,c), q(c,e), r(e,a,c),
q(d,f), **r(a,a,d)**,
r(e,e,g)}



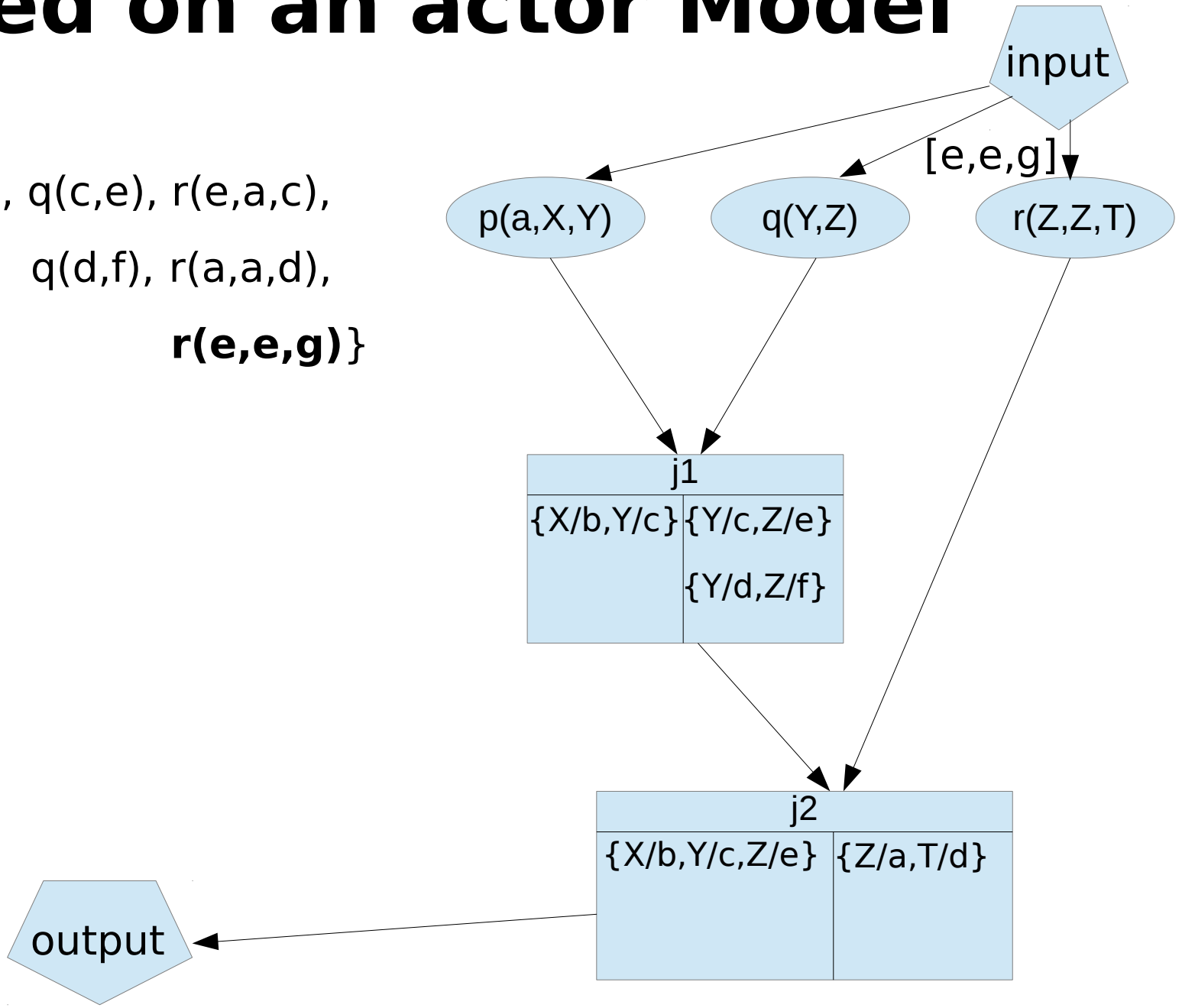
Split θ-subsumption based on an actor Model

B = {p(a,b,c), q(c,e), r(e,a,c),
q(d,f), **r(a,a,d)**,
r(e,e,g)}



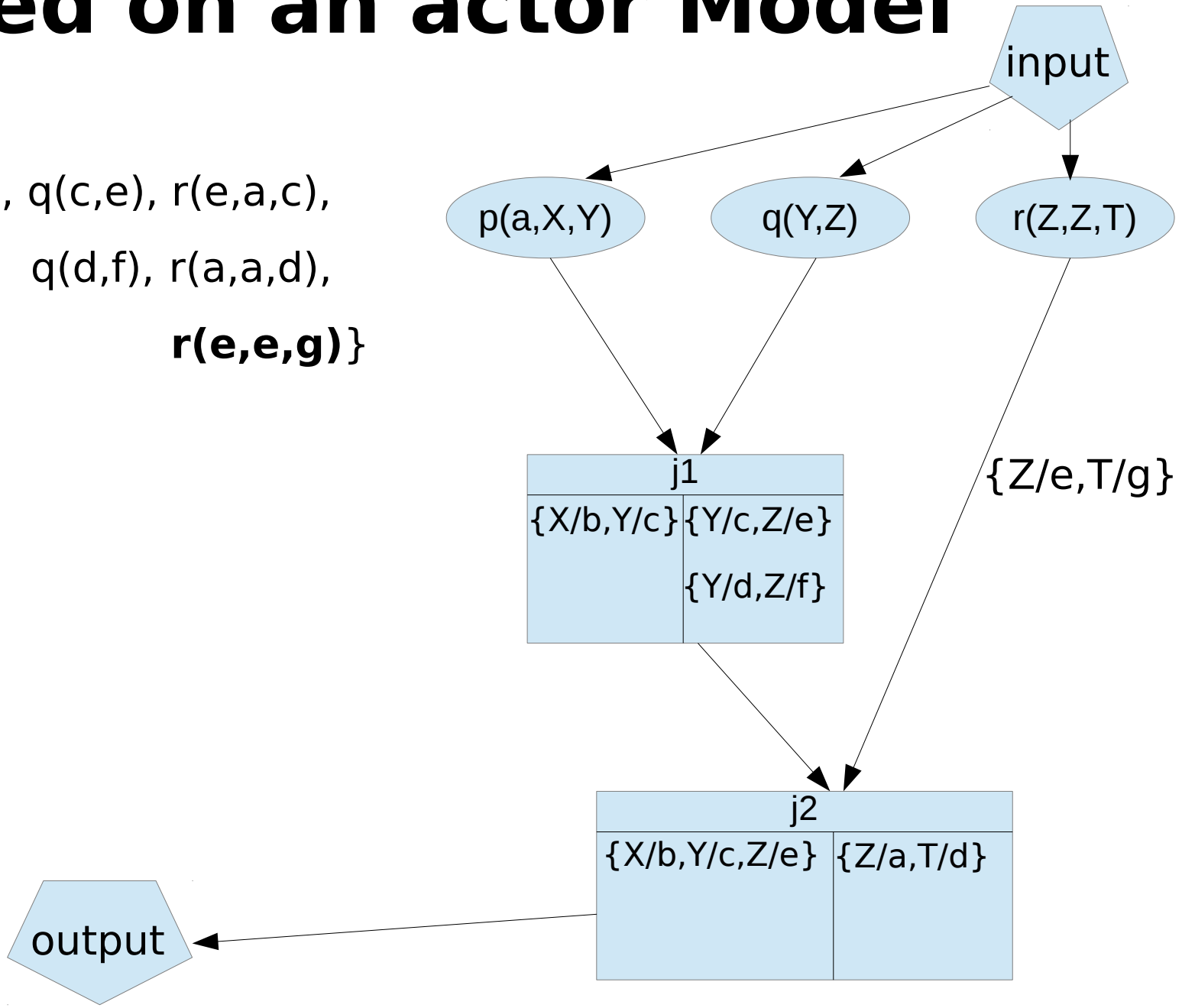
Split θ-subsumption based on an actor Model

B = {p(a,b,c), q(c,e), r(e,a,c),
q(d,f), r(a,a,d),
r(e,e,g)}



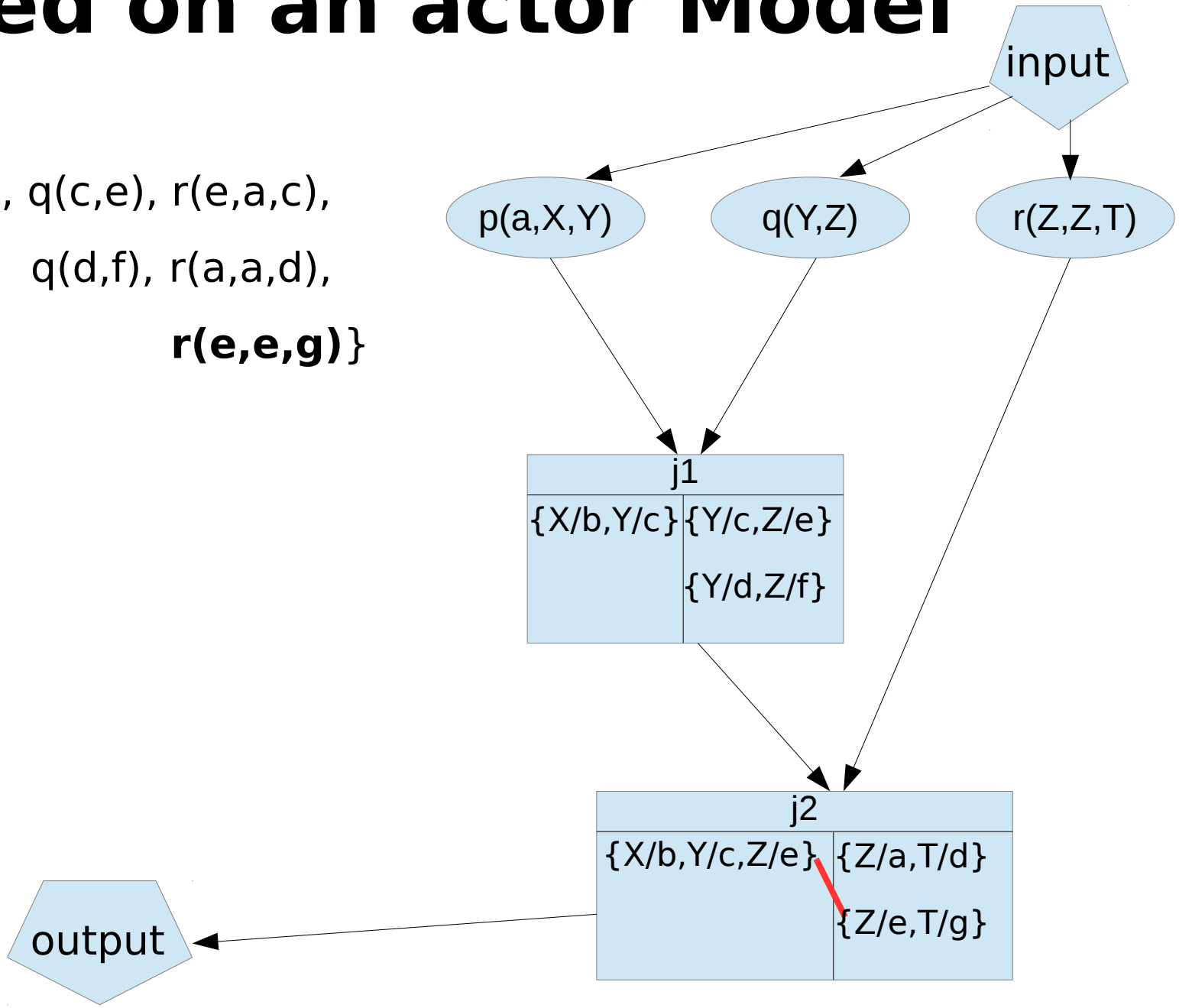
Split θ-subsumption based on an actor Model

B = {p(a,b,c), q(c,e), r(e,a,c),
q(d,f), r(a,a,d),
r(e,e,g)}



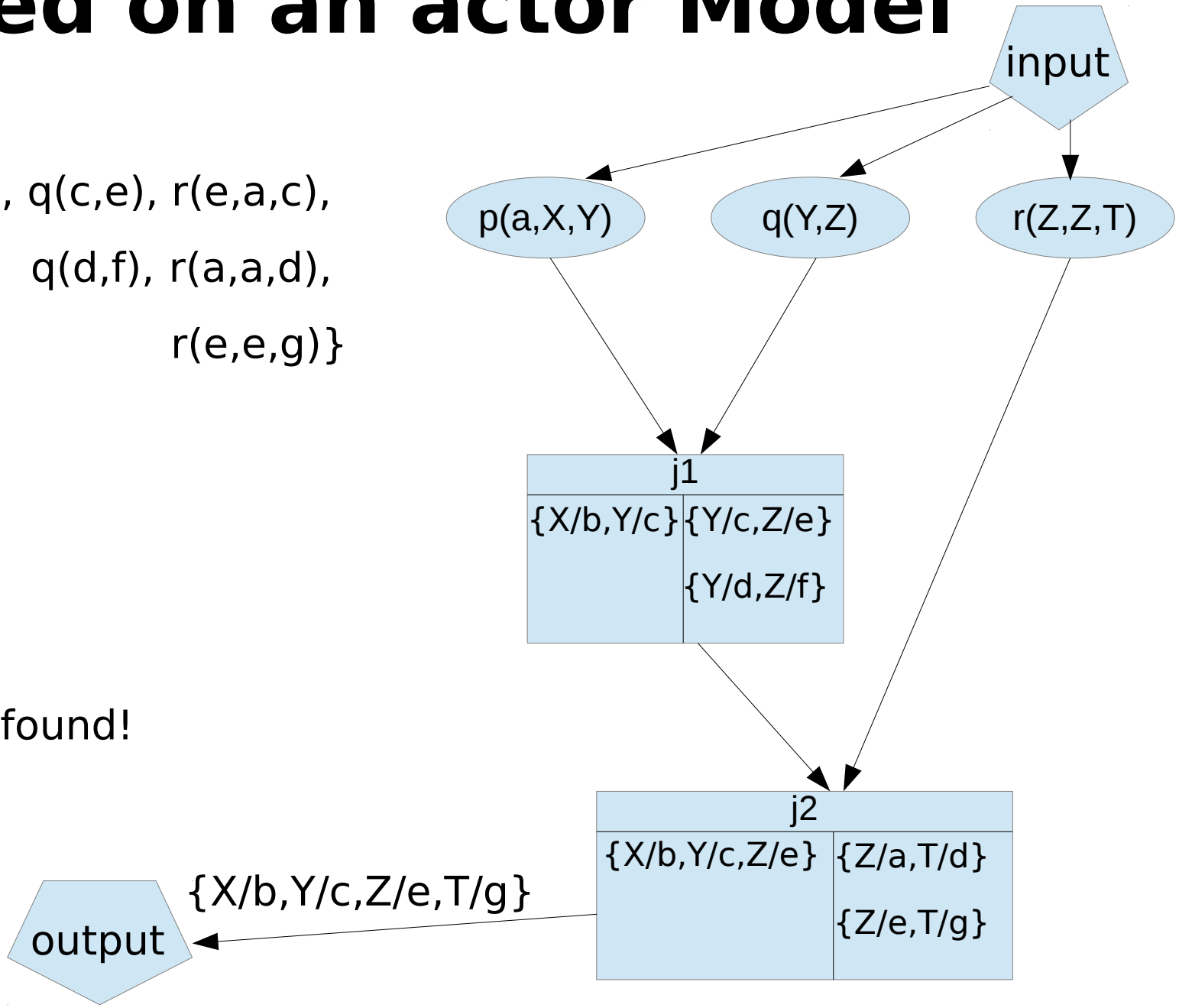
Split θ-subsumption based on an actor Model

B = {p(a,b,c), q(c,e), r(e,a,c),
q(d,f), r(a,a,d),
r(e,e,g)}



Split θ-subsumption based on an actor Model

B = {p(a,b,c), q(c,e), r(e,a,c),
q(d,f), r(a,a,d),
r(e,e,g)}



One solution found!

Experiments

Implementation

- We used the Akka framework, whith the Scala langage

Implementation

- We used the Akka framework, whith the Scala langage

- Parallel implementation on a single machine

- Distributed implementation on clusters

First experiments

Environment:

- 64bit Linux system
- Intel Core i7-5600U (**2 hyperthreaded cores** running at 2.6GHz)
- 8GB memory
- Akka 2.4.4 and Scala 2.11.7

First experiments

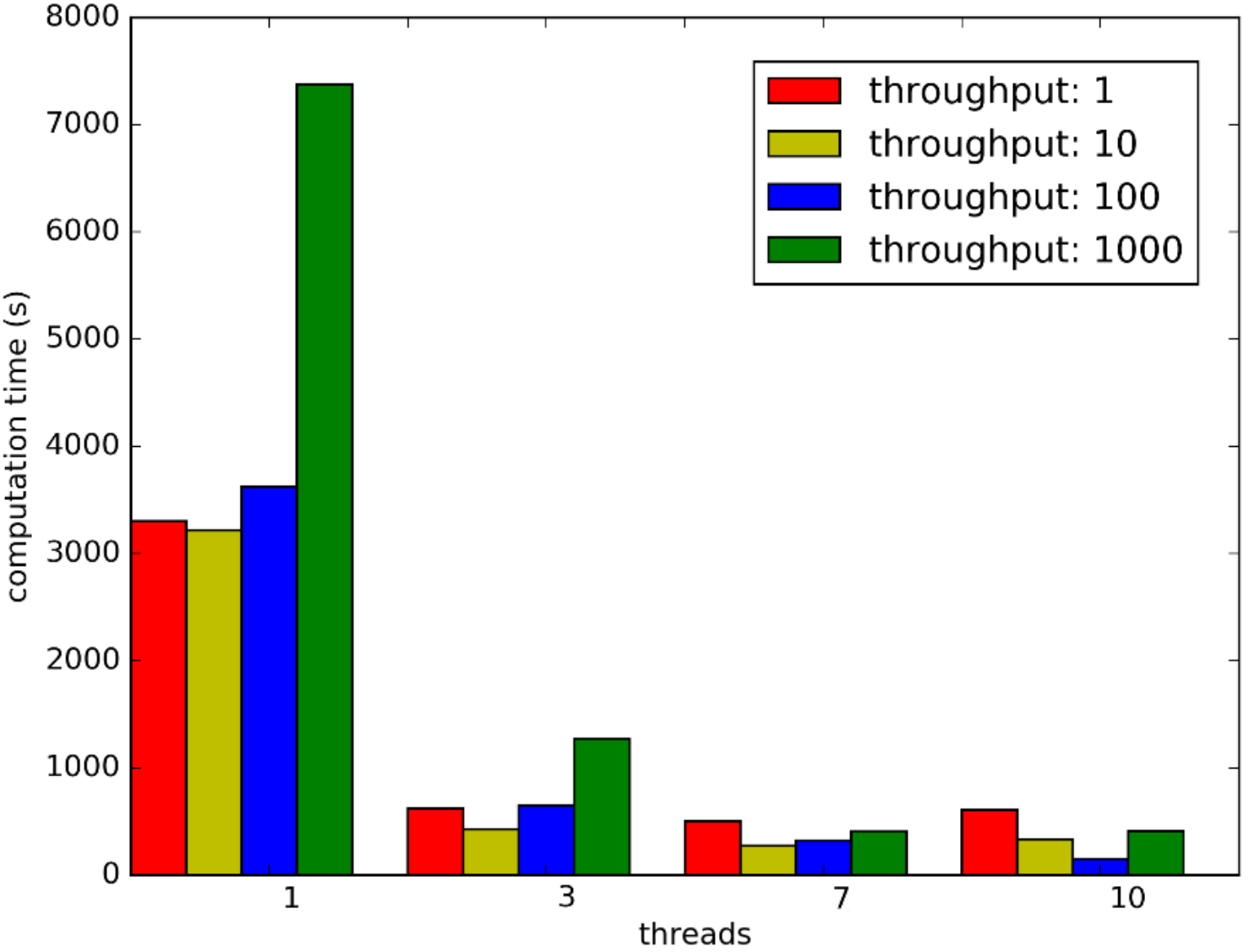
Environment:

- 64bit Linux system
- Intel Core i7-5600U (**2 hyperthreaded cores** running at 2.6GHz)
- 8GB memory
- Akka 2.4.4 and Scala 2.11.7

Data:

- Truncated from the data generated for *Subsumer* (Santos et al. 2010) : generated from instances of the PT problem
- 30 literals in subsumer clause, 200 literals in subsumee clause
- 4 distinct predicate symbols (arity of 2-3)
- 8 distinct variable symbols

First experiments



Ongoing experiments

- Distributed implementation of our model on the Grid’5000 testbed

supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations

Conclusions and perspectives

Conclusion

- New model of parallelized and distributed θ subsumption
- Parallel experiments
- Distributed experiments (work in progress)

Persectives

- Distributed experiments (work in progress)
- Classic ILP optimizations (clause partitioning, linked variable analysis, …)
- Alternative actor model, where the actors are built from the subsumee’s literals instead of the subsumer’s literals

Persectives

- Distributed experiments (work in progress)
- Classic ILP optimizations (clause partitioning, linked variable analysis, …)
- Alternative actor model, where the actors are built from the subsumee’s literals instead of the subsumer’s literals

Thank you