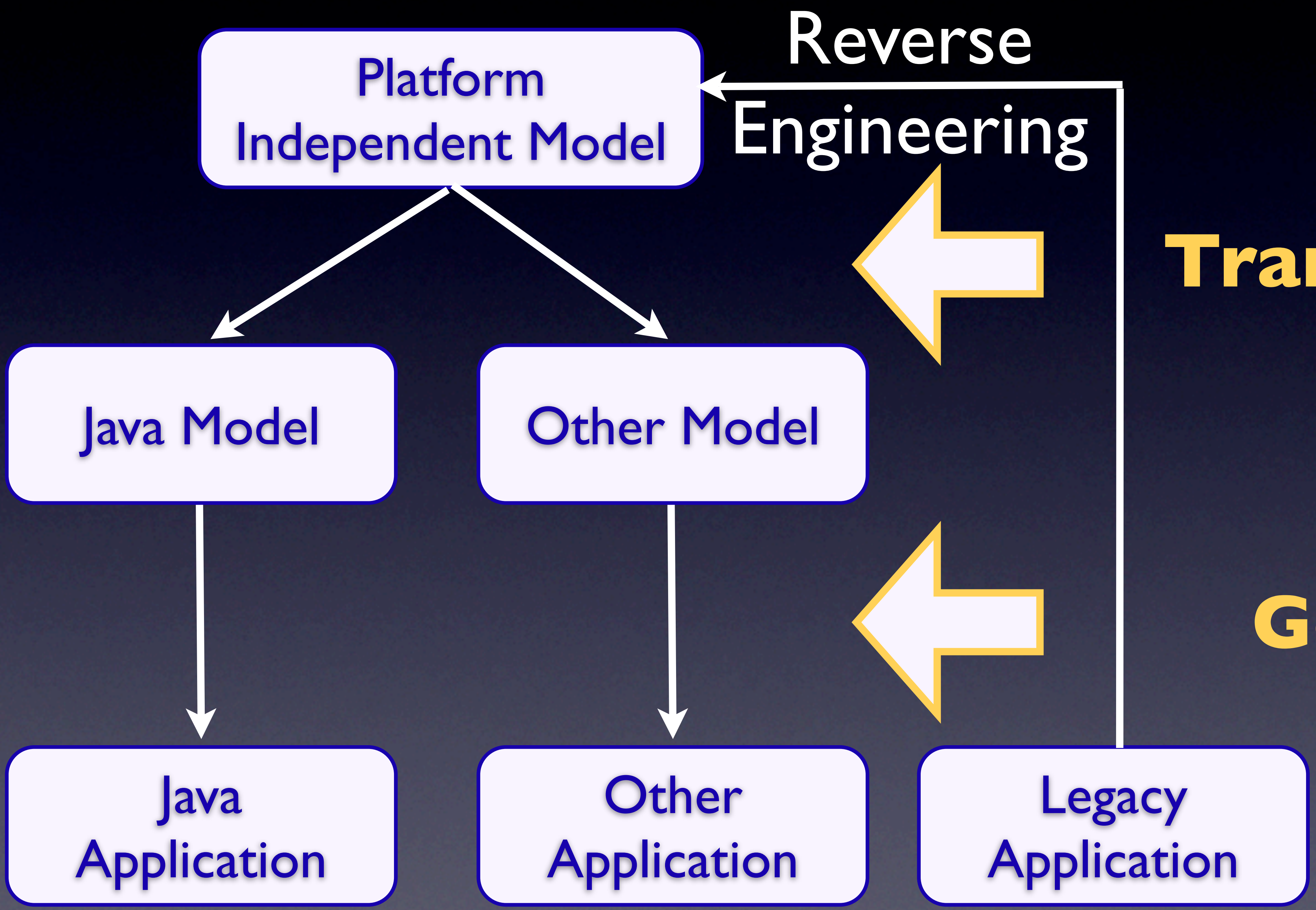
Static Type Checking of Model Transformation Programs

Zoltán Ujhelyi

Fault Tolerant Systems Research Group
Department of Measurement and Information Systems

Model-Driven Software Development



Model Transformation (M2M)

Code Generation (M2T)

Transformation program

```
rule fireTransition(in Tr) = seq{
  forall P with
    find sourcePlace(Tr, P) do
        choose To with find
          placeWithToken(P,To) do
            call removeToken(P);

...
}
```

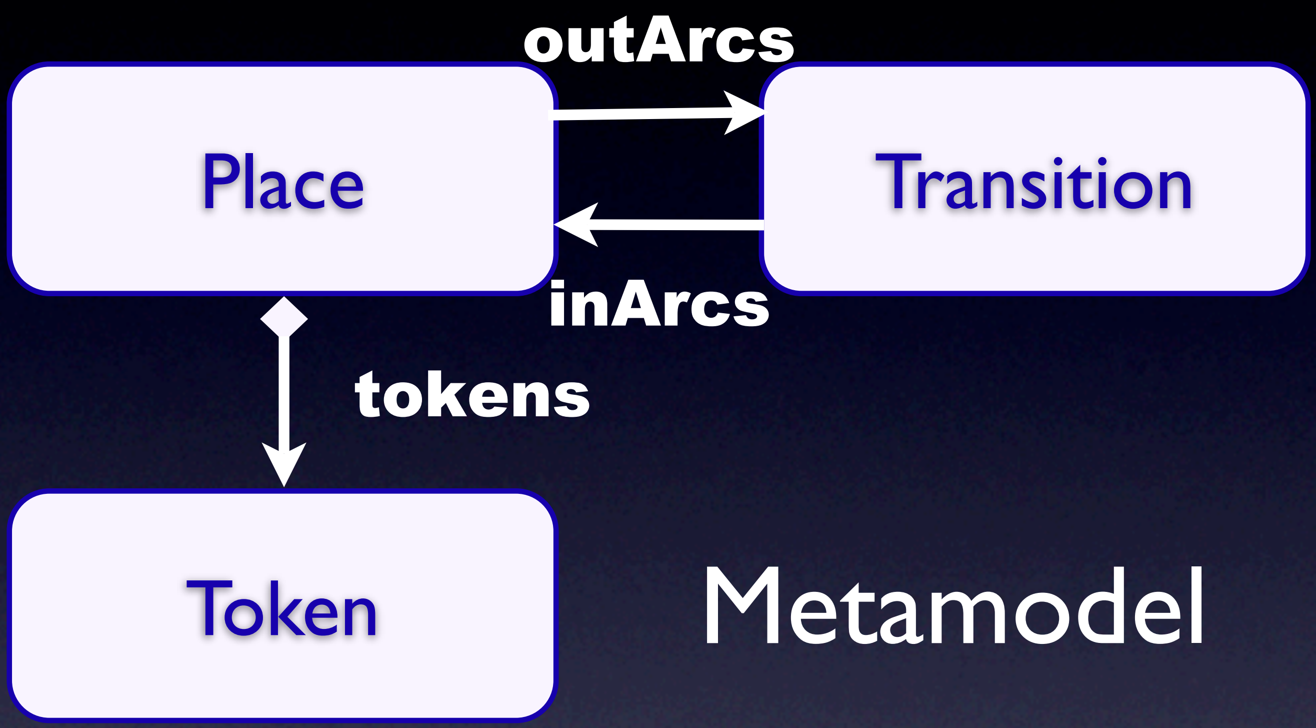
Transformation program

```
rule fireTransition(in Tr) = seq{
  forall P with
    find sourcePlace(Tr, P) do
        choose To with find
          placeWithToken(P,To) do
            call removeToken(P);

...
}
```

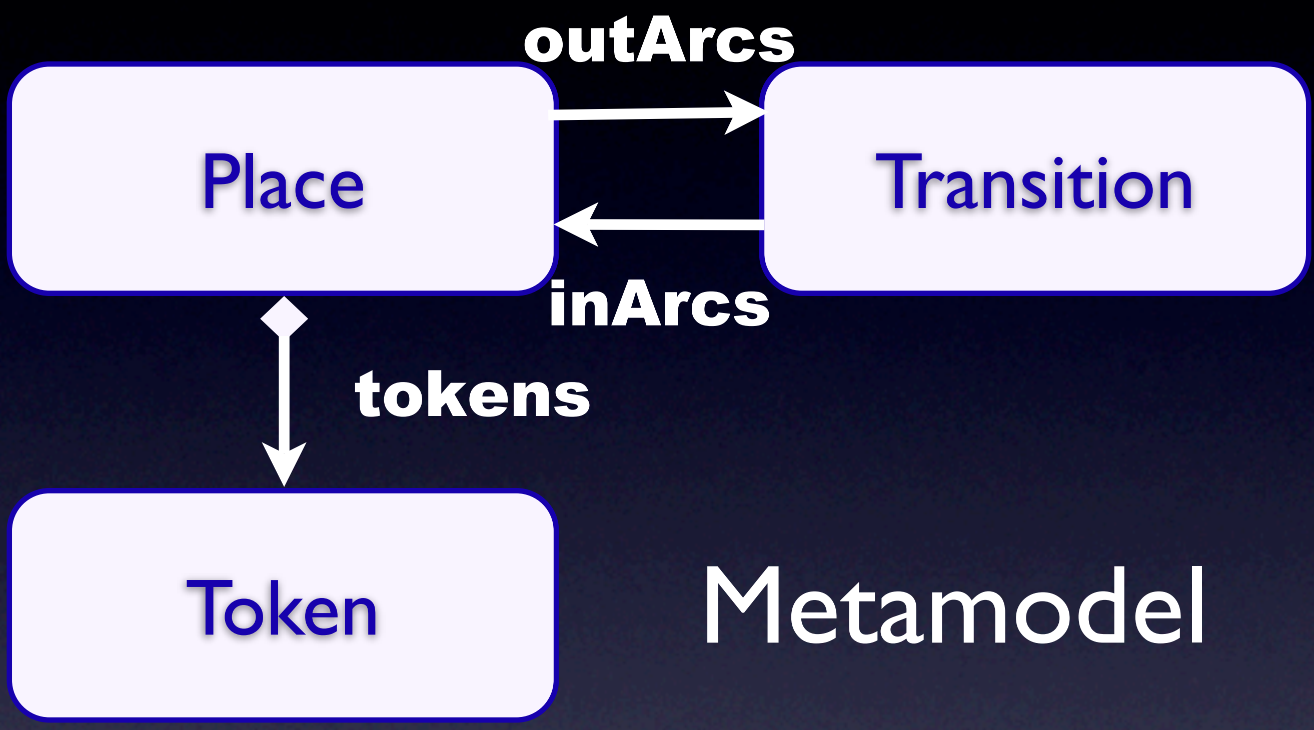
- Special program
  - Input/output models
  - Metamodels
  - GT rules

Metamodeling



Metamodel

METAMODELING



Place

Transition

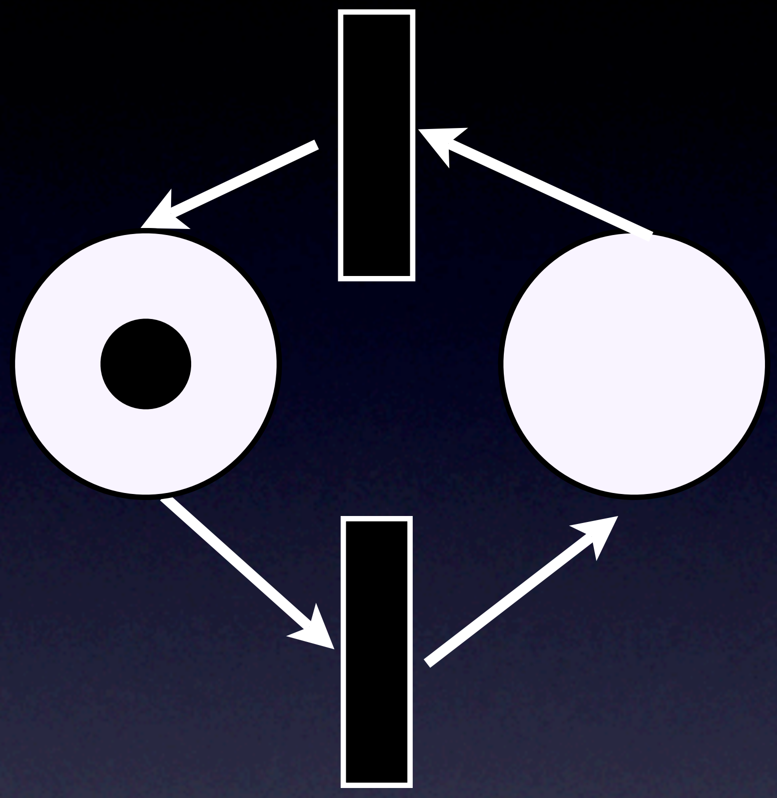
Token

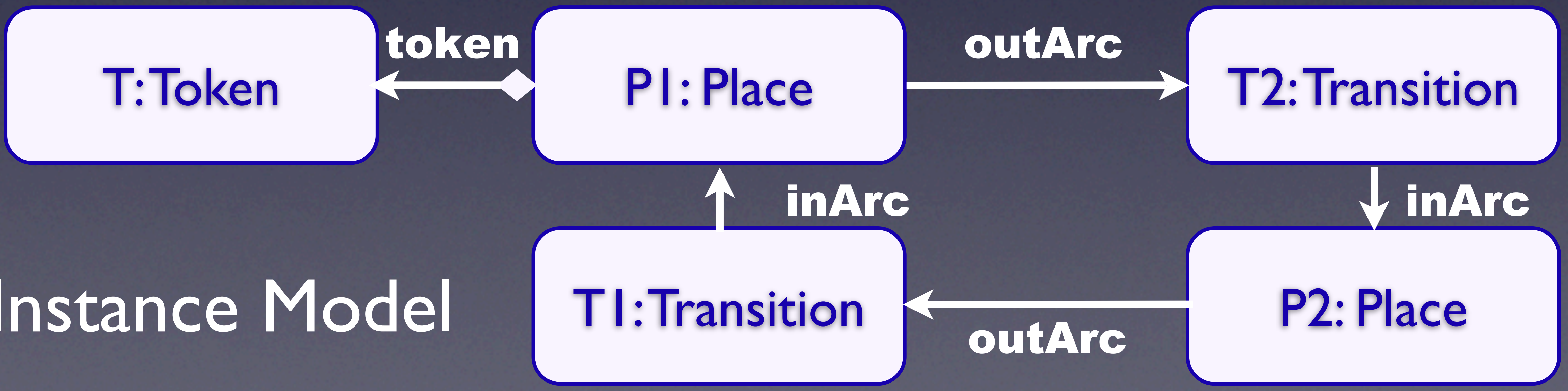
outArcs

inArcs

tokens

Metamodel





T:Token

P1: Place

T2: Transition

T1: Transition

P2: Place

token

outArc

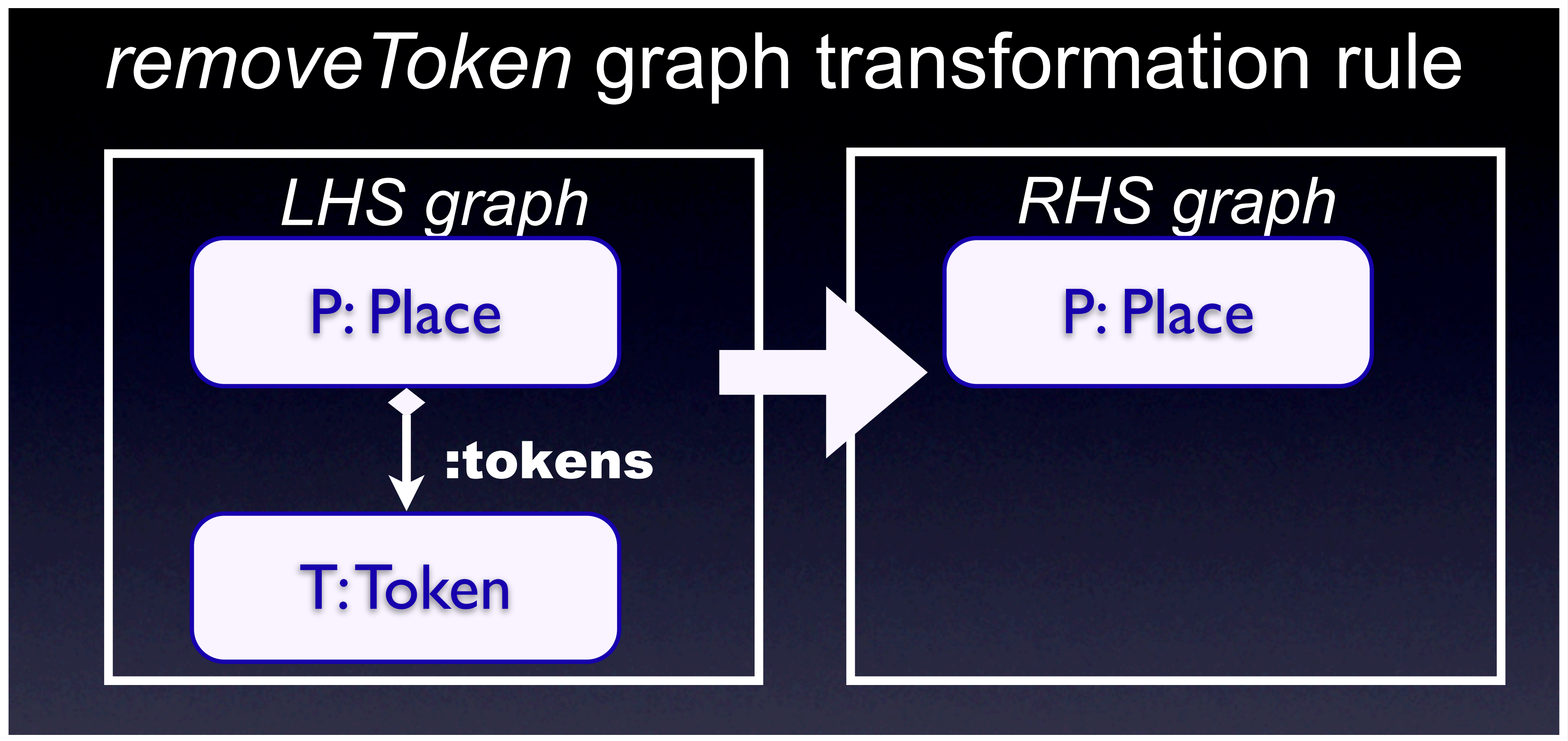
inArc

inArc

outArc

Instance Model

Graph Transformation



removeToken graph transformation rule

LHS graph

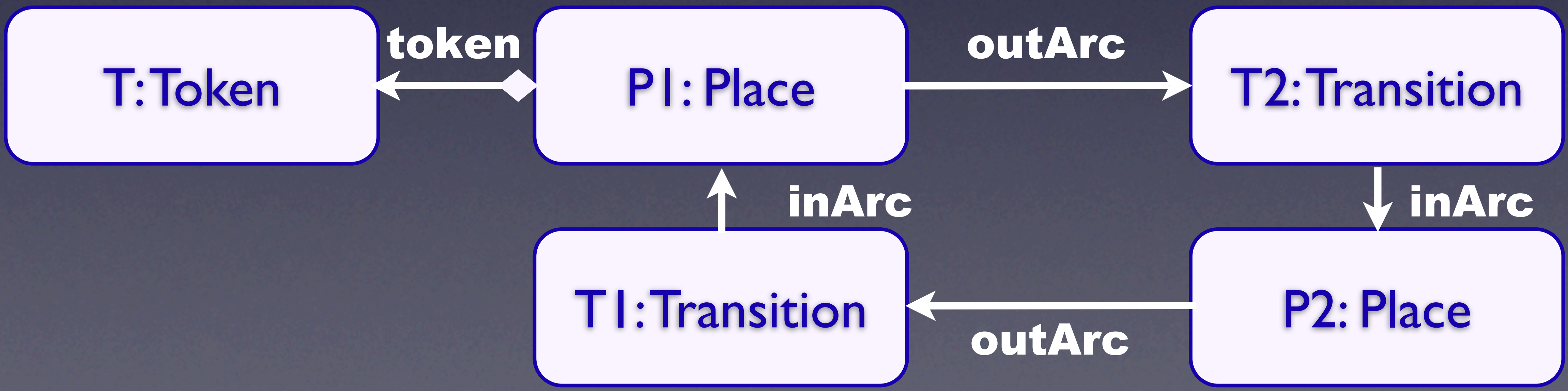
P: Place

:tokens

T: Token

RHS graph

P: Place



T: Token

token

P1: Place

outArc

T2: Transition

inArc

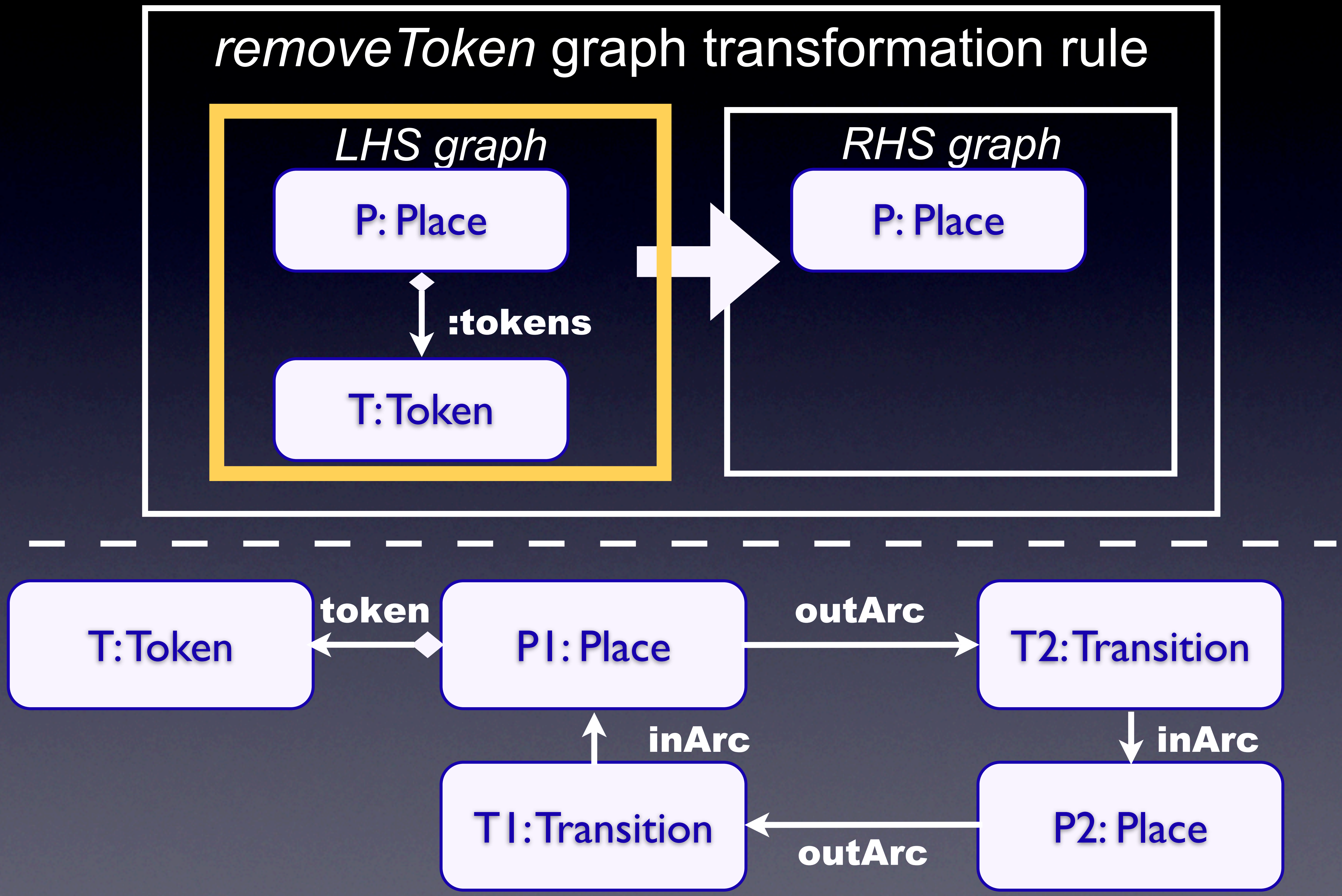
inArc

T1: Transition

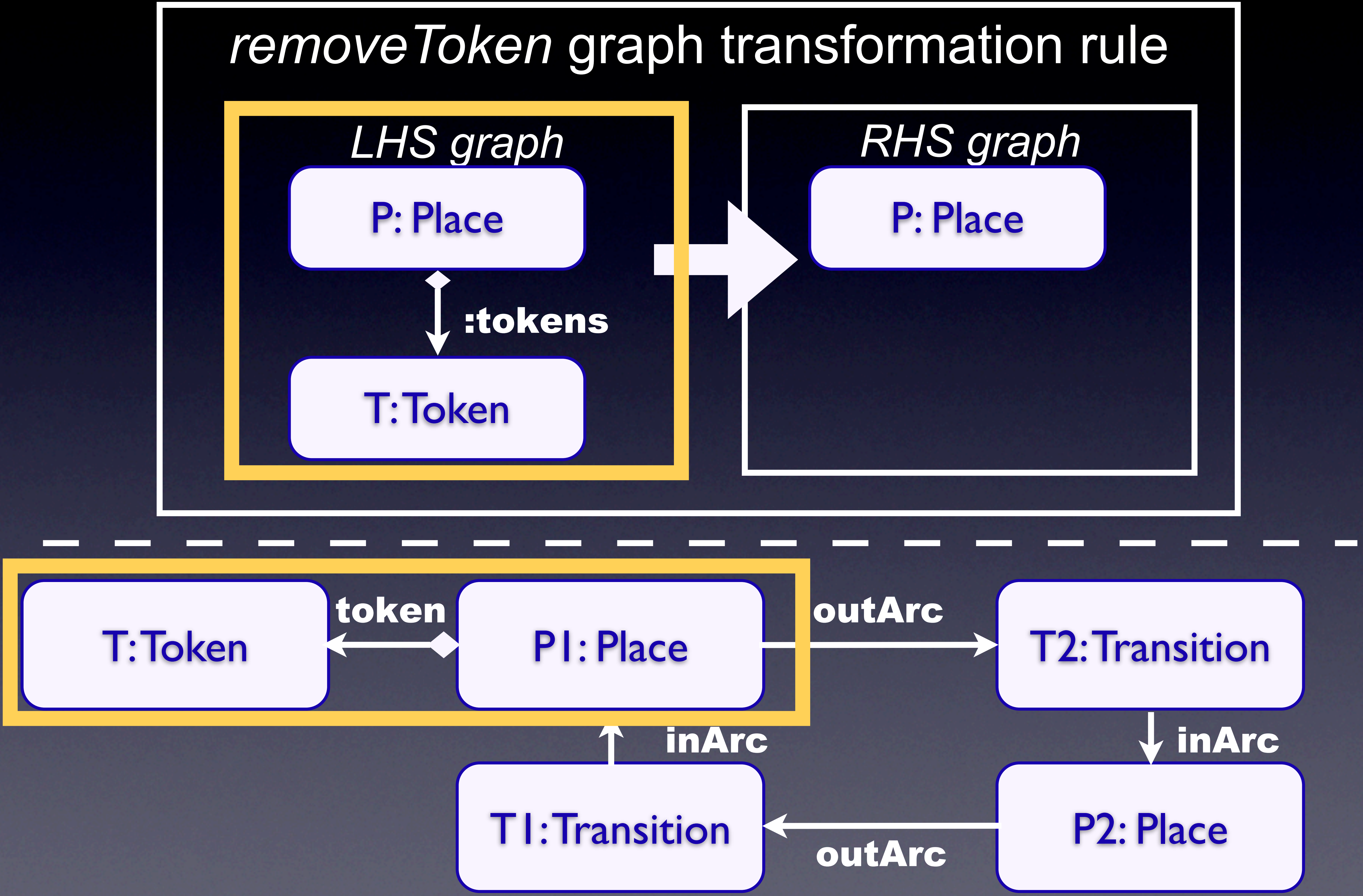
outArc

P2: Place

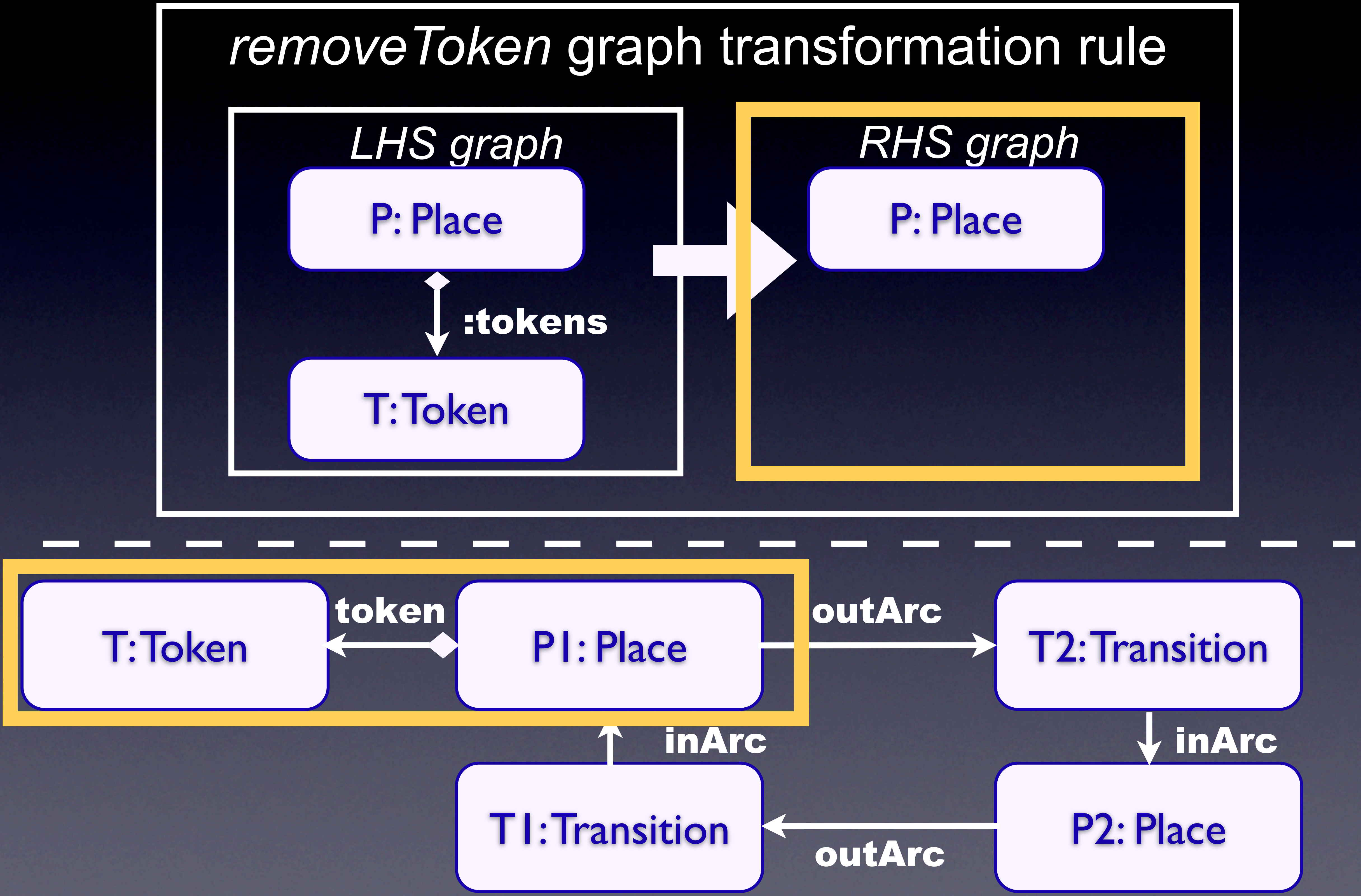
Graph Transformation



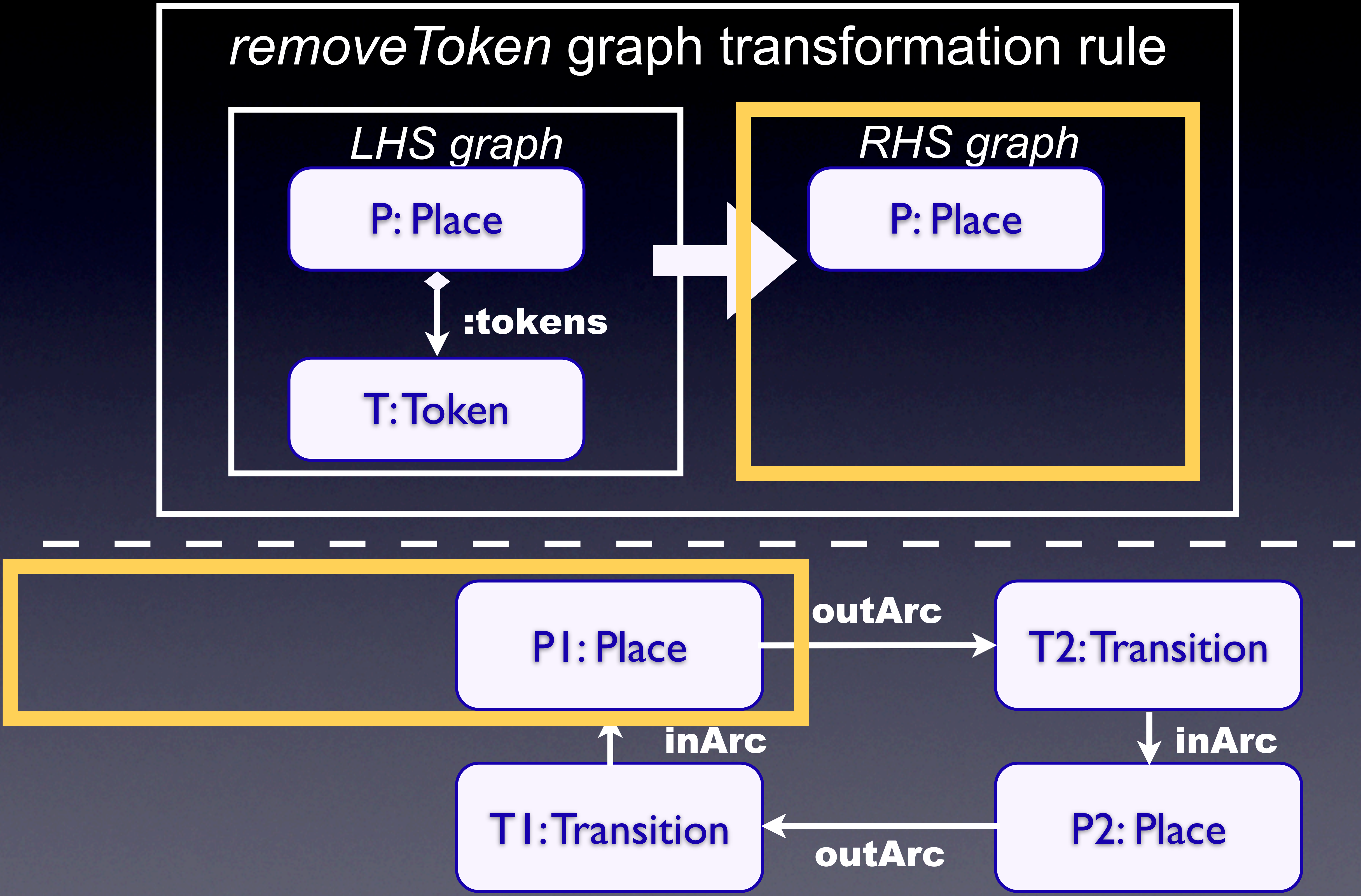
Graph Transformation



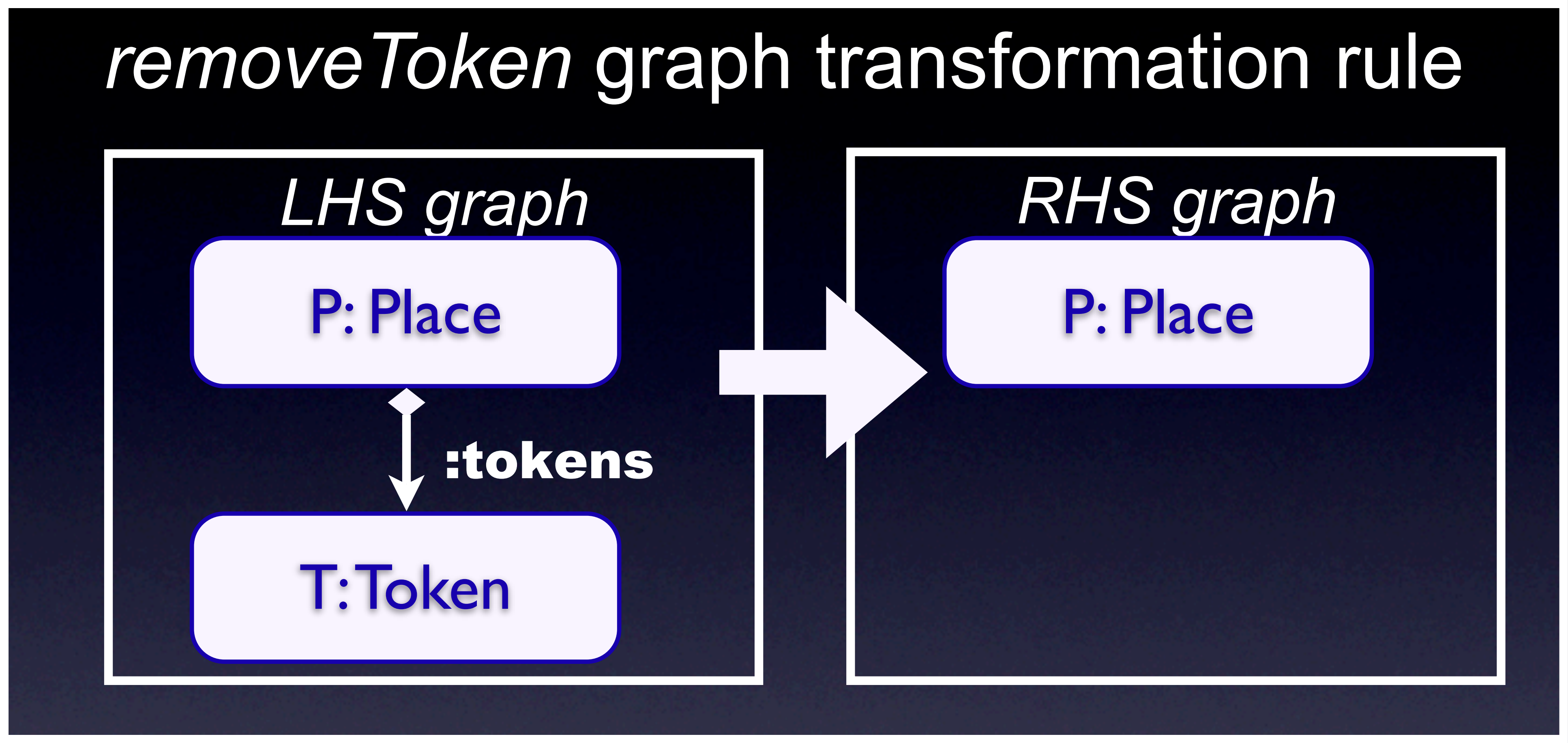
Graph Transformation

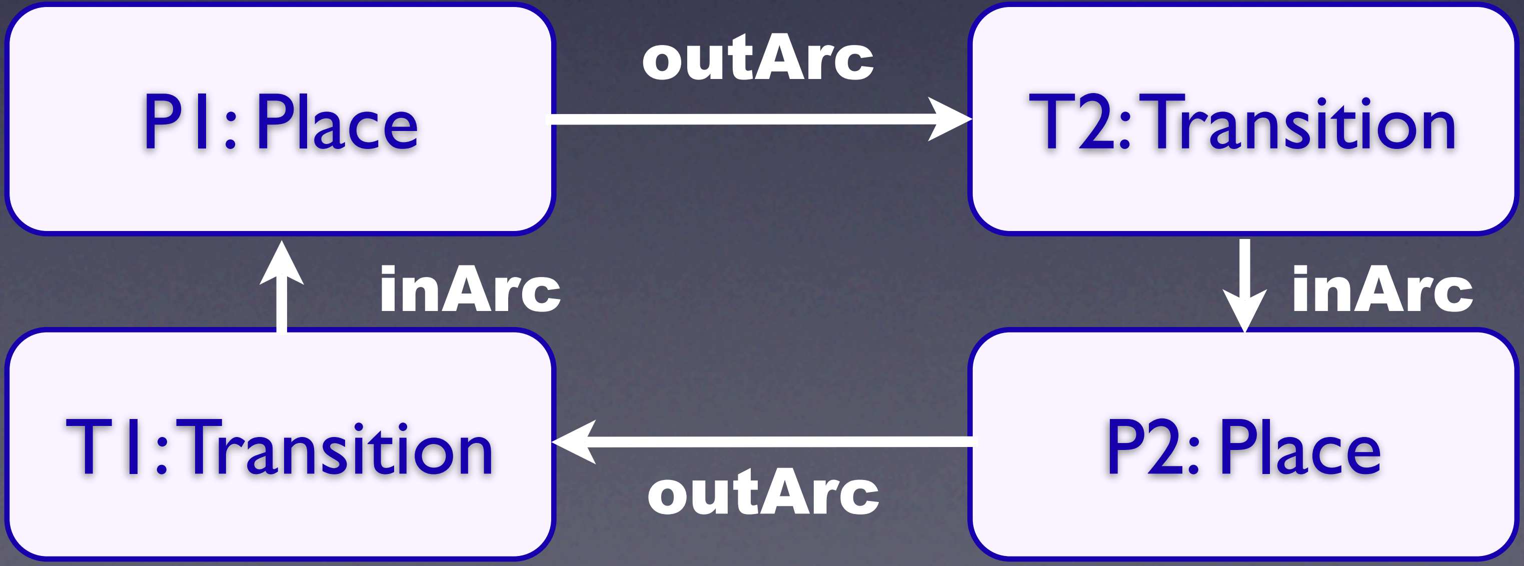


Graph Transformation



Graph Transformation





Transformation program

```
rule fireTransition(in Tr) = seq{
  forall P with
    find sourcePlace(Tr, P) do
        choose To with find
          placeWithToken(P,To) do
            call removeToken(P);

...
}
```

Transformation program

```
rule fireTransition(in Tr) = seq{
  forall P with
    find sourcePlace(Tr, P) do
        choose To with find
          placeWithToken(P,To) do
            call removeToken(P);

...
}
```

- Developer errors
- Manual validation is hard

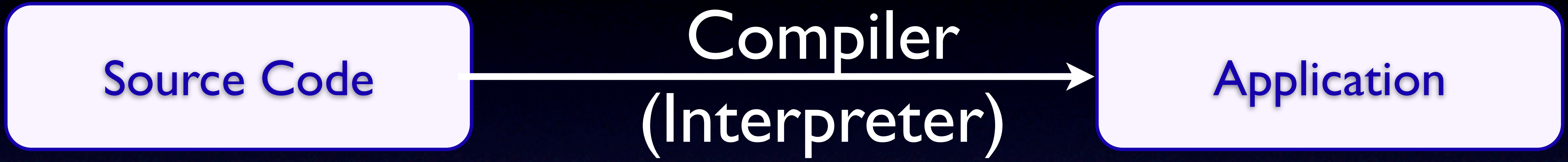
Transformation program

```
rule fireTransition(in Tr) = seq{
  forall P with
    find sourcePlace(Tr, P) do
        choose To with find
          placeWithToken(P,Tr) do
            call removeToken(P);

...
}
```

- Developer errors
- Manual validation is hard

Static Analysis



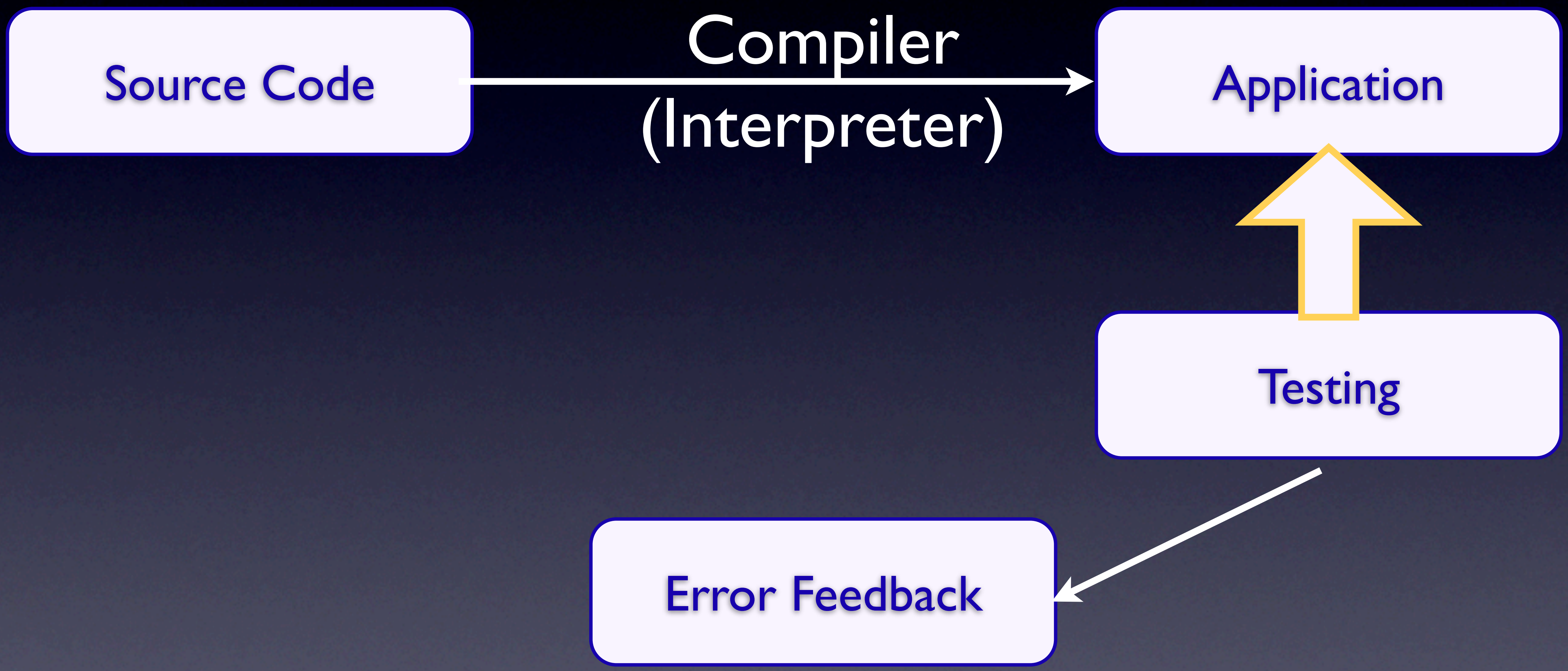
Source Code

Compiler

(Interpreter)

Application

Static Analysis


Source Code

Compiler

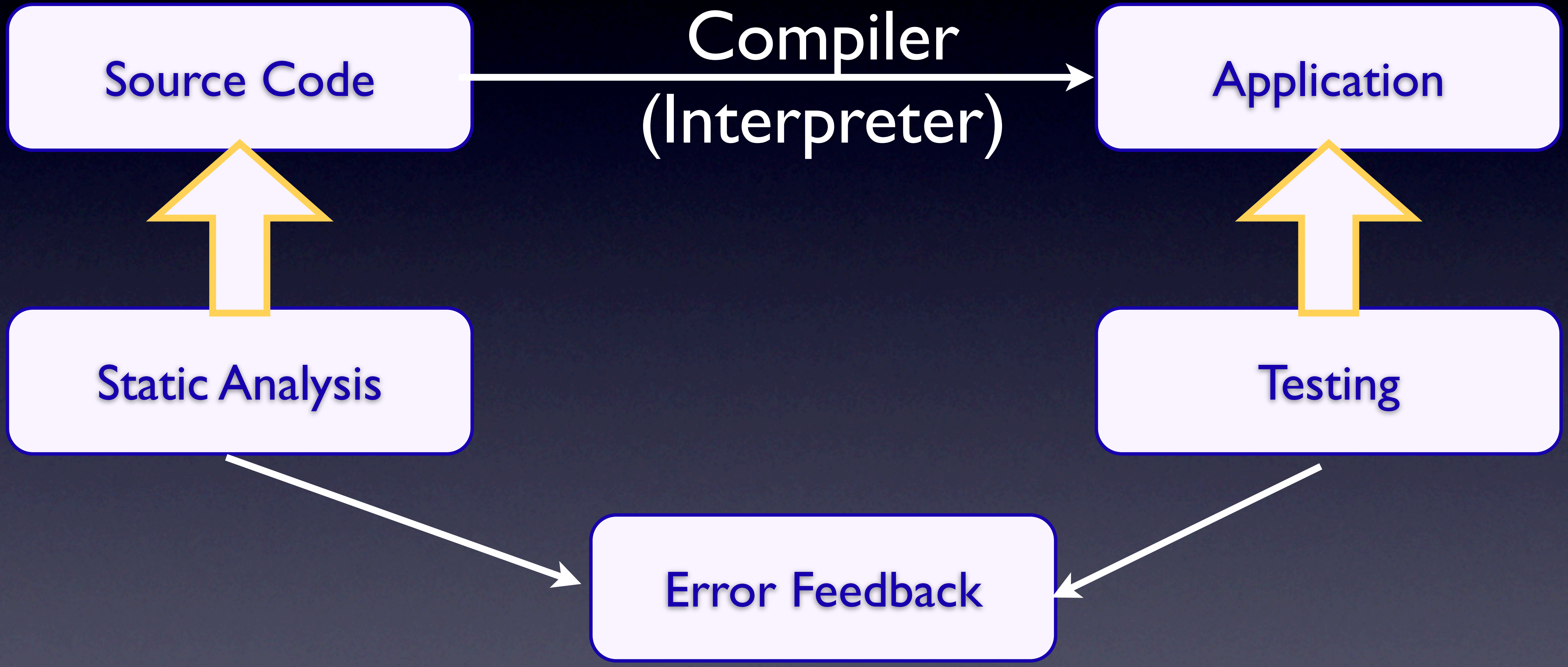
(Interpreter)

Application

Testing

Error Feedback

Static Analysis



Source Code

Compiler (Interpreter)

Application

Static Analysis

Testing

Error Feedback

Static Type Checking

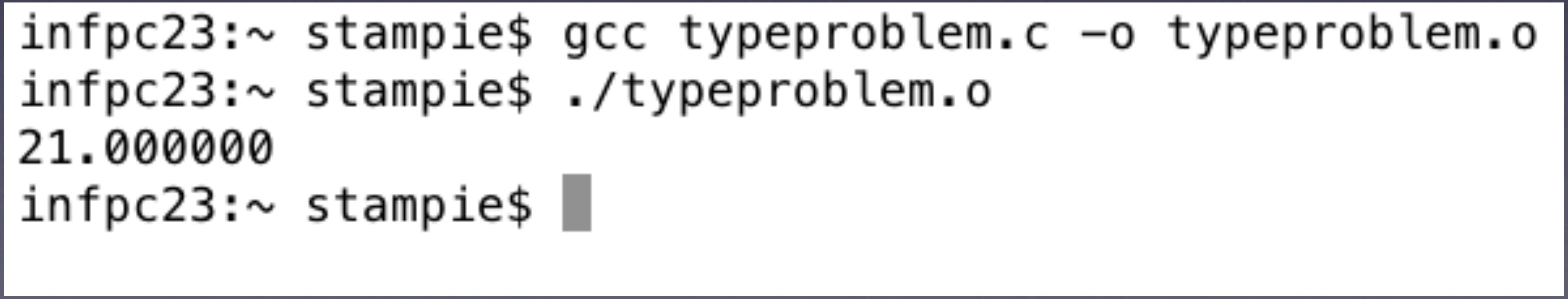
```
int main() {
  float x = addTen(11.9f);
  printf("%f\n", x);
  return 0;
}

int addTen(int n) {
  return n + 10;
}
```

Static Type Checking

```c
int main() {
  float x = addTen(11.9f);
  printf("%f\n", x);
  return 0;
}

int addTen(int n) {
  return n + 10;
}
```

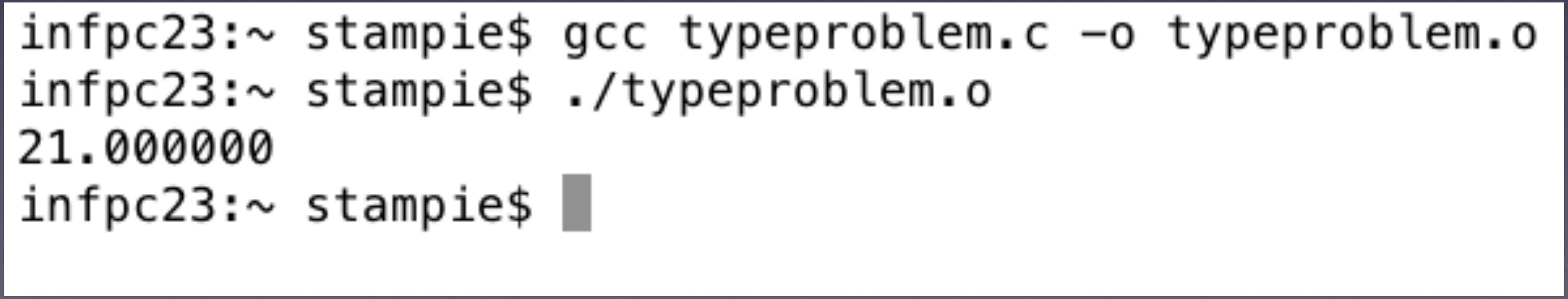


```
infpc23:~ stampie$ gcc typeproblem.c -o typeproblem.o
infpc23:~ stampie$ ./typeproblem.o
21.000000
infpc23:~ stampie$ 
```

Static Type Checking

```
int main() {
   float x = addTen(11.9f);
   printf("%f\n", x);
   return 0;
}

int addTen(int n) {
   return n + 10;
}
```



```
infpc23:~ stampie$ gcc typeproblem.c -o typeproblem.o
infpc23:~ stampie$ ./typeproblem.o
21.000000
infpc23:~ stampie$ 
```

Goals

- Static analyser framework
  - general
    - multiple properties
- **Type checking**
  - solver: constraint satisfaction problems
- Implementation
  - Extending the VIATRA2 framework

Type Checking, as a CSP

```
...
rule fireTransition(in Tr) = seq{
  forall P with
   find sourcePlace(Tr, P) do
    choose To with find
        placeWithToken(P,Tr) do
       call removeToken(P);
...
}
...
```

Type Checking, as a CSP

```
...
rule fireTransition(in Tr) = seq{
  forall P with
   find sourcePlace(Tr, P) do
    choose To with find
        placeWithToken(P,Tr) do
       call removeToken(P);
...
}
...
```

Place 1

Transition 2

Token 3

Type Checking, as a CSP

```
...
rule fireTransition(in Tr) = seq{
  forall P with
   find sourcePlace(Tr, P) do
    choose To with find
         placeWithToken(P,Tr) do
        call removeToken(P);
...
}
...
```

Place ❶

Transition ❷

Token ❸

Tr: {❶❷❸}

P: {❶❷❸}

To: {❶❷❸}

Type Checking, as a CSP

```
...
rule fireTransition(in Tr)     ng{
  forall P with
   find sourcePlace(
    choose To with fi
        placeWithToken(P,Tr) do
       call removeToken(P);
...
}
...
```

Pattern call:
P: Place
Tr: Token

Place ❶

Transition ❷

Token ❸

Tr: {❶❷❸}
P: {❶❷❸}
To: {❶❷❸}

Type Checking, as a CSP

```
...
rule fireTransition(in Tr)        g{
  forall P with
   find sourcePlace(
    choose To with fi
        placeWithToken(P,Tr) do
        call removeToken(P);
...
}
...
```

Pattern call:
P: Place
Tr: Token

Place ①

Transition ②

Token ③

Tr: {③}
P: {①}
To: {①②③}

Type Checking, as a CSP

```
...
rule fireTransition
  forall P with
   find sourcePlace(Tr, P) do
    choose To with find
        placeWithToken(P,Tr) do
       call removeToken(P);
...
}
...
```

Pattern call:
Tr: Transition
P: Place

Place ❶

Transition ❷

Token ❸

Tr: {❸}
P: {❶}
To: {❶❷❸}

Type Checking, as a CSP

```
...
rule fireTransition
  forall P with
   find sourcePlace(Tr, P) do
    choose To with find
        placeWithToken(P,Tr) do
       call removeToken(P);
...
}
...
```

Pattern call:
Tr: Transition
P: Place

Place ❶

Transition ❷

Token ❸

Tr: {}

P: {❶}

To: {❶❷❸}

Type Checking, as a CSP

```
...
rule fireTransition
  forall P with
   find sourcePlace(Tr, P) do
    choose To with find
        placeWithToken(P,Tr) do
       call removeToken(P);
...
}
...
```

Pattern call:
Tr: Transition
P: Place

Place ❶

Transition ②

③

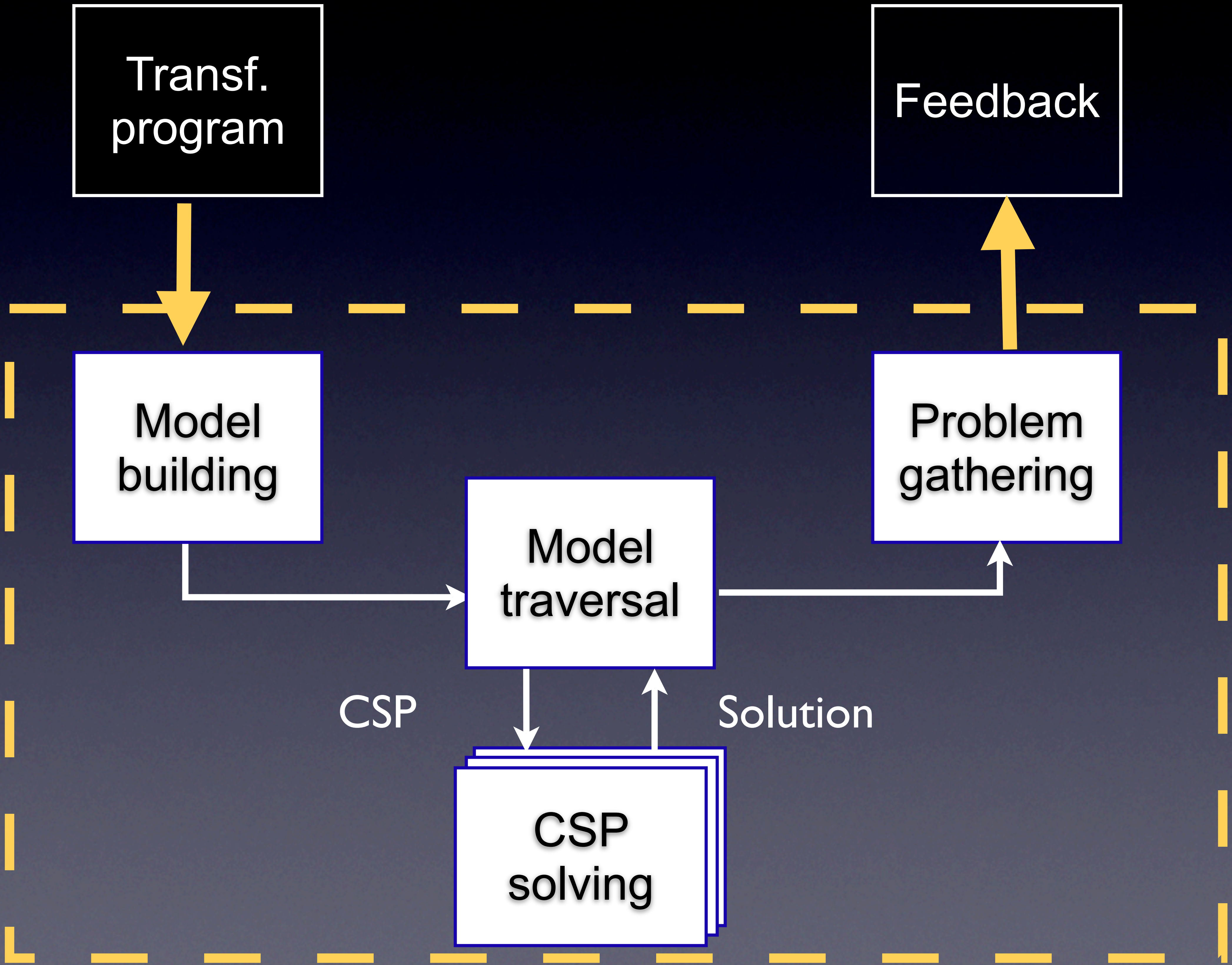
No solution:
domain of Tr is empty

Tr: {}

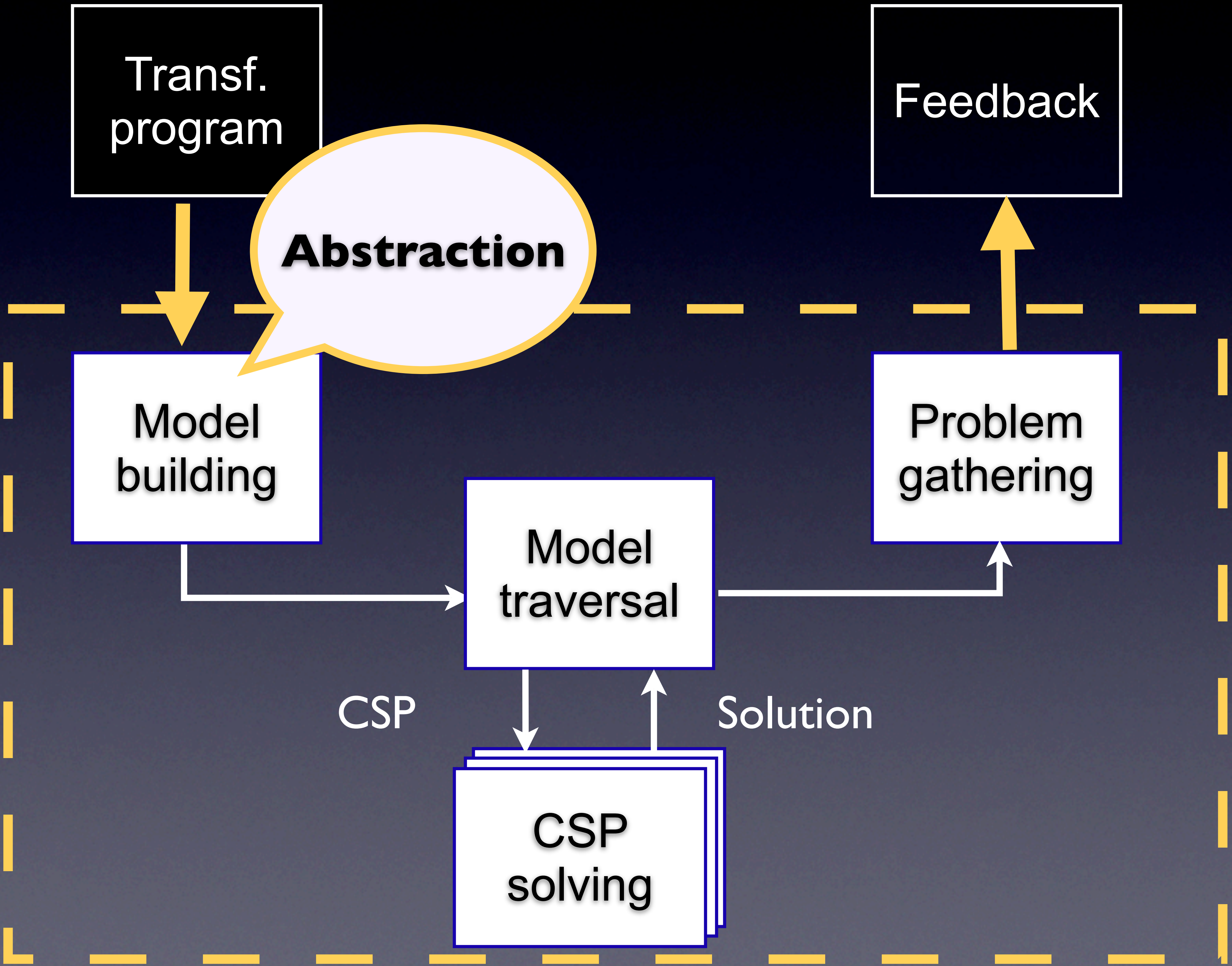
P: {❶}

To: {❶②③}

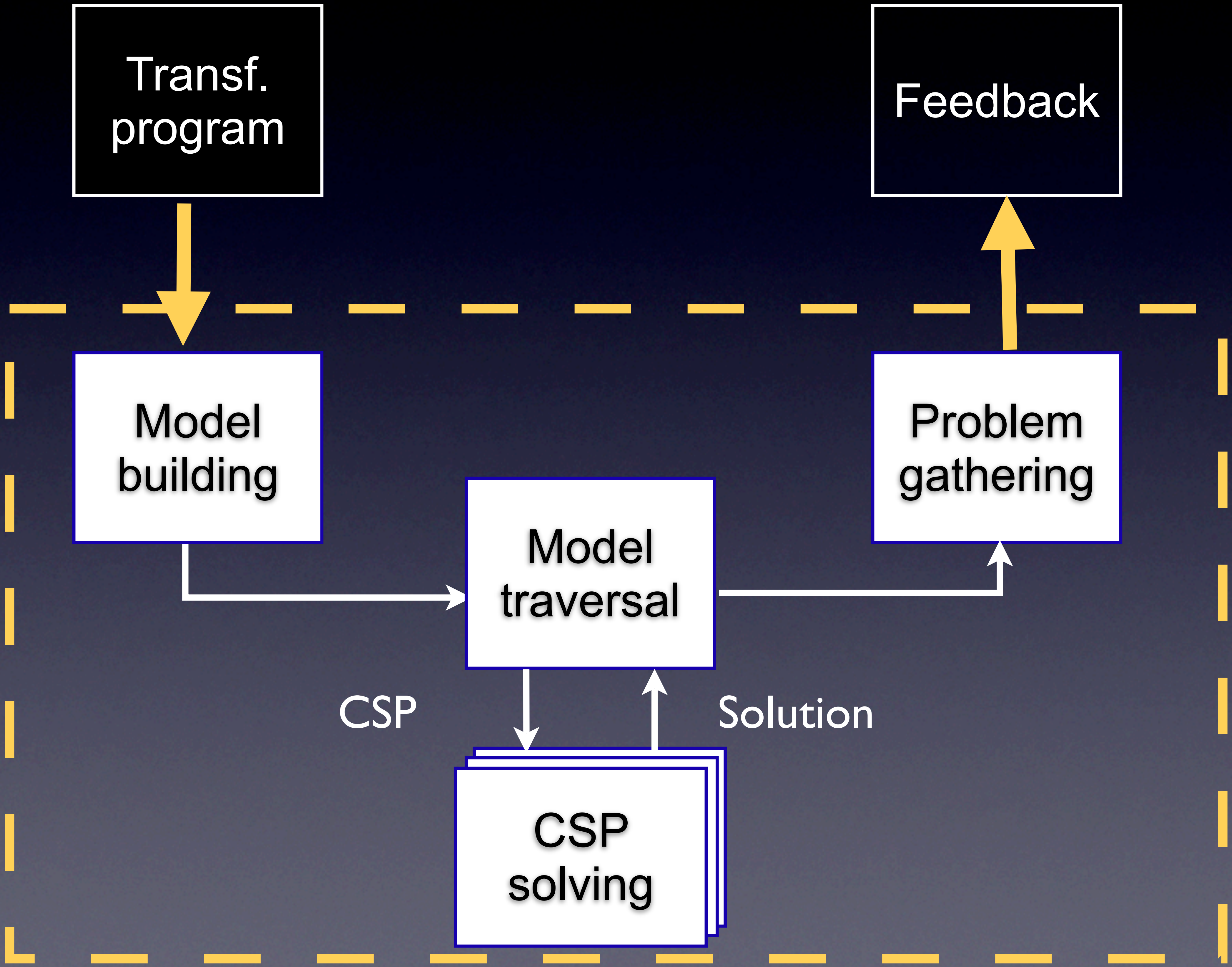
Solution


Transf. program
Feedback
Model building
Problem gathering
Model traversal
CSP
Solution
CSP solving

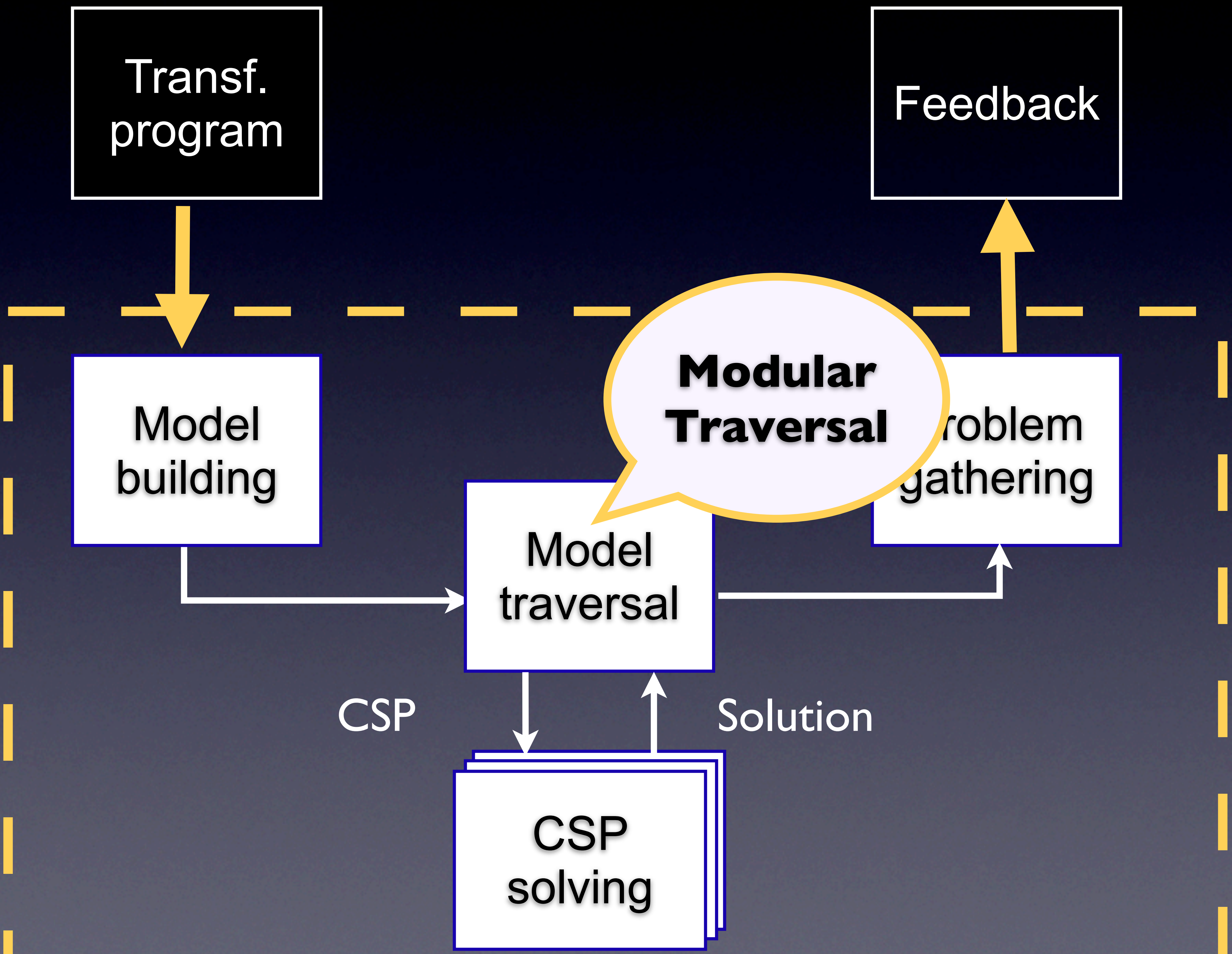
Solution


Transf. program
Abstraction
Feedback
Model building
Problem gathering
Model traversal
CSP
Solution
CSP solving

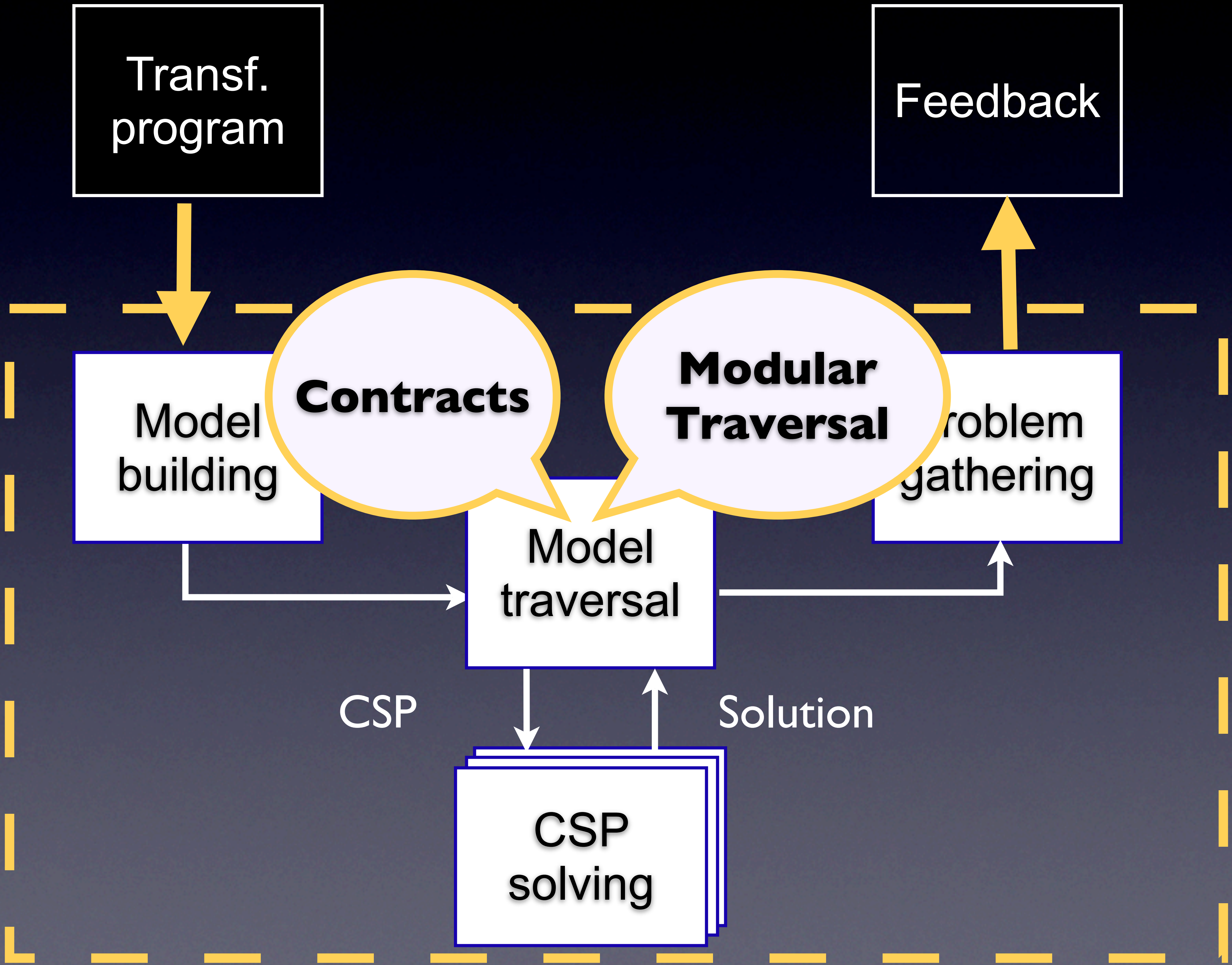
Solution


Transf. program
Feedback
Model building
Problem gathering
Model traversal
CSP
Solution
CSP solving

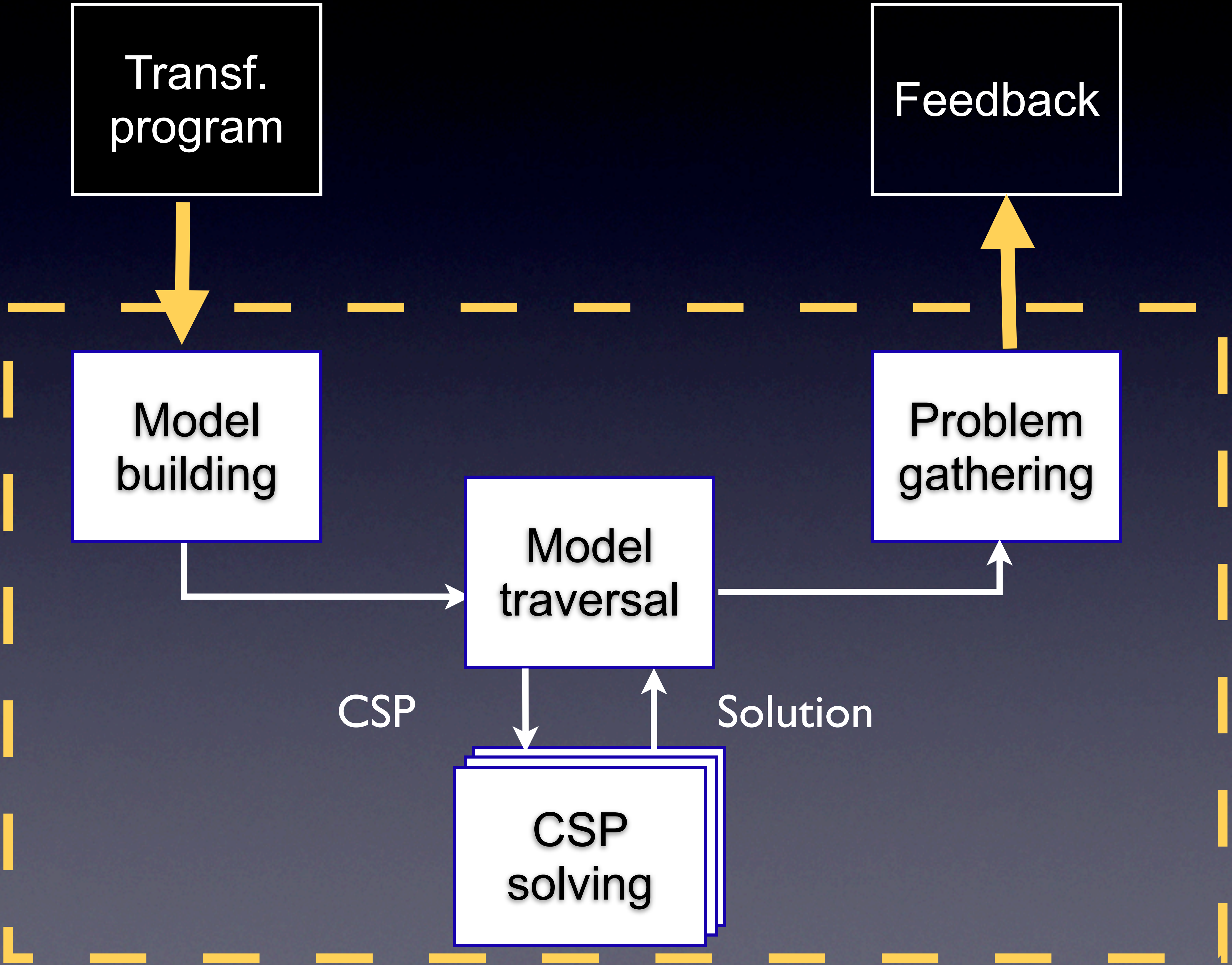
Solution


Transf. program
Feedback
Model building
Modular Traversal
Problem gathering
Model traversal
CSP
Solution
CSP solving

Solution


Transf. program
Feedback
Model building
Contracts
Modular Traversal
roblem gathering
Model traversal
CSP
Solution
CSP solving

Solution



Transf. program

Feedback

Model building

Problem gathering

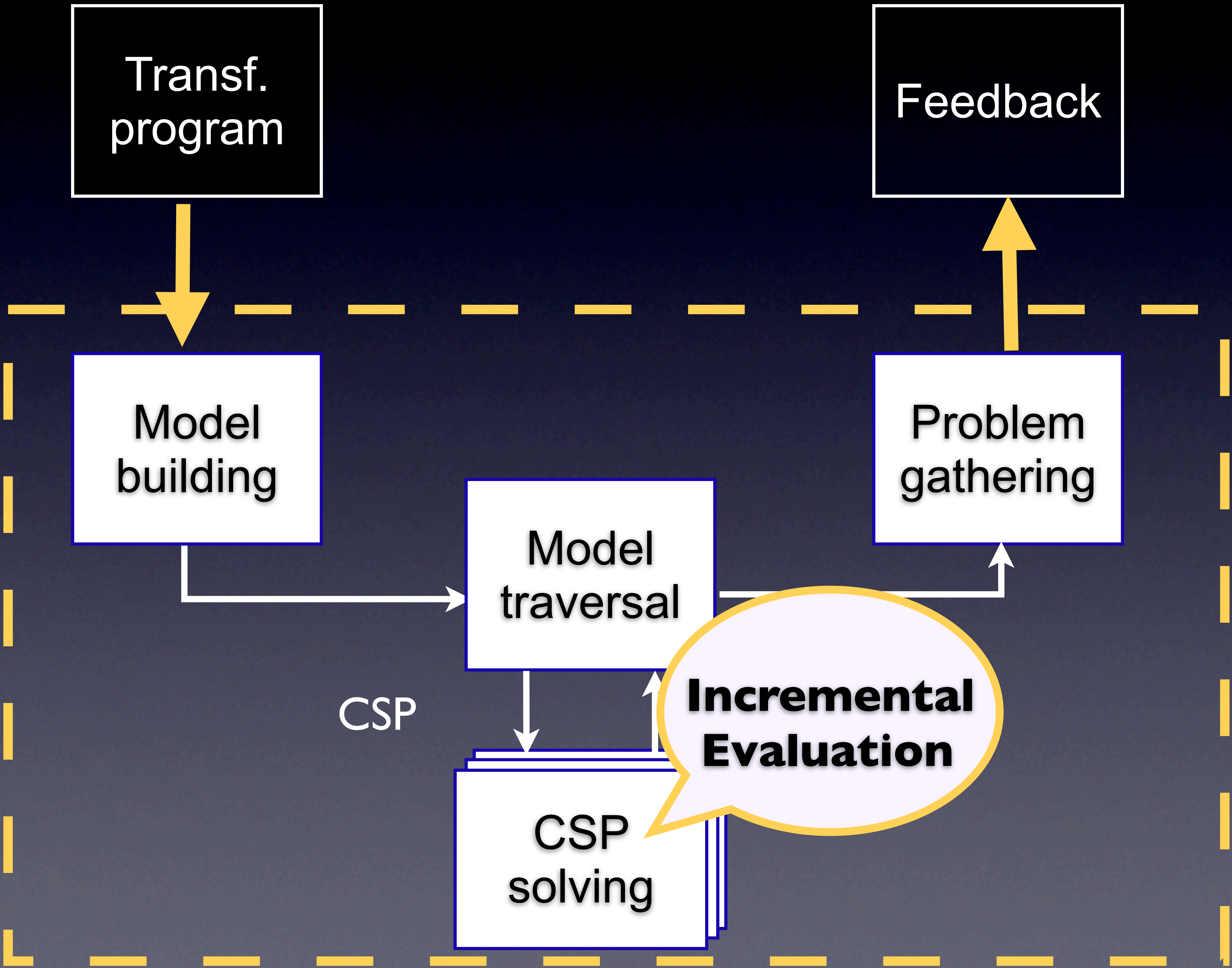
Model traversal

CSP

Solution

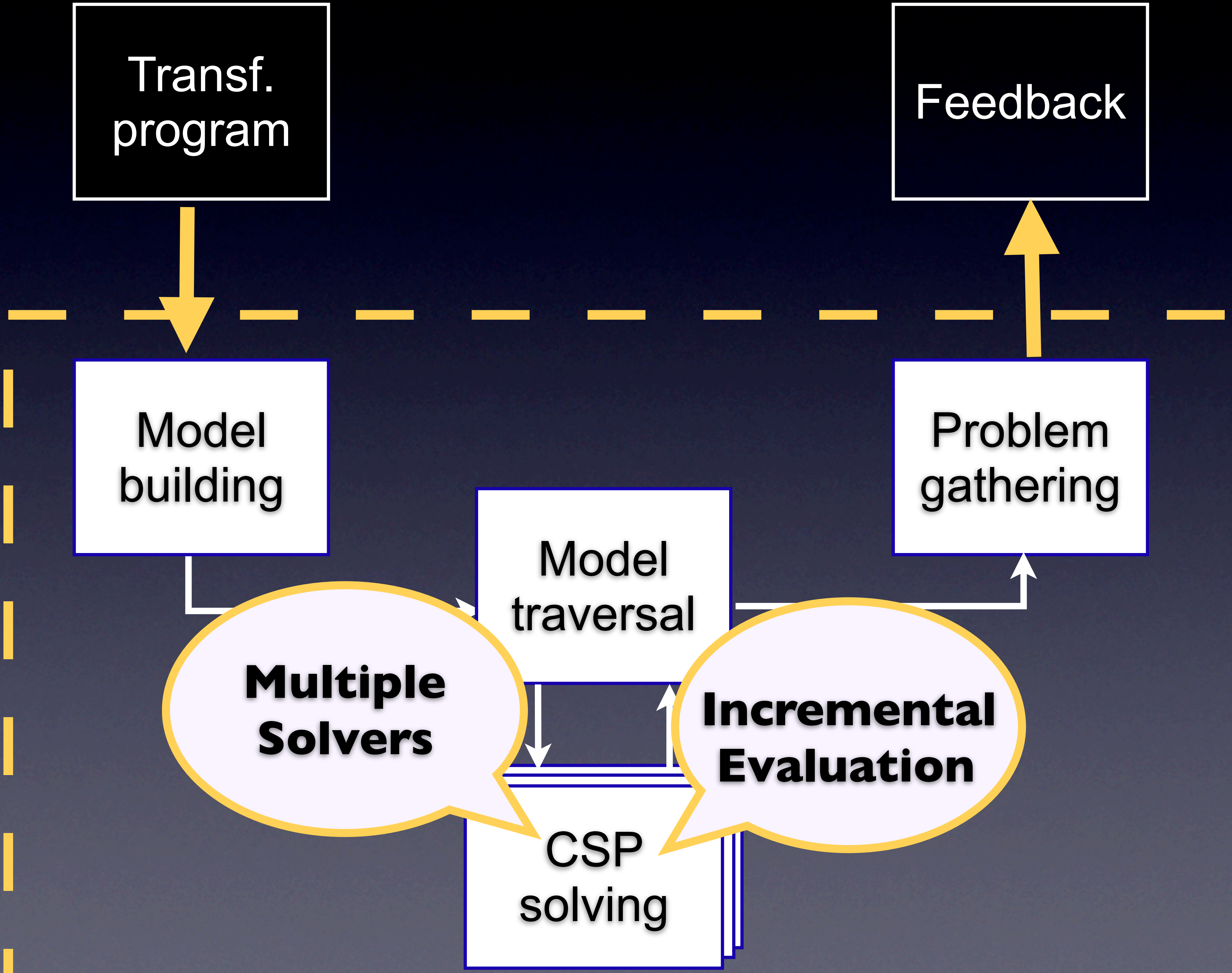
CSP solving

Solution



Transf. program
Feedback
Model building
Model traversal
Problem gathering
CSP
CSP solving
Incremental Evaluation

Solution



Transf. program

Feedback

Model building

Model traversal

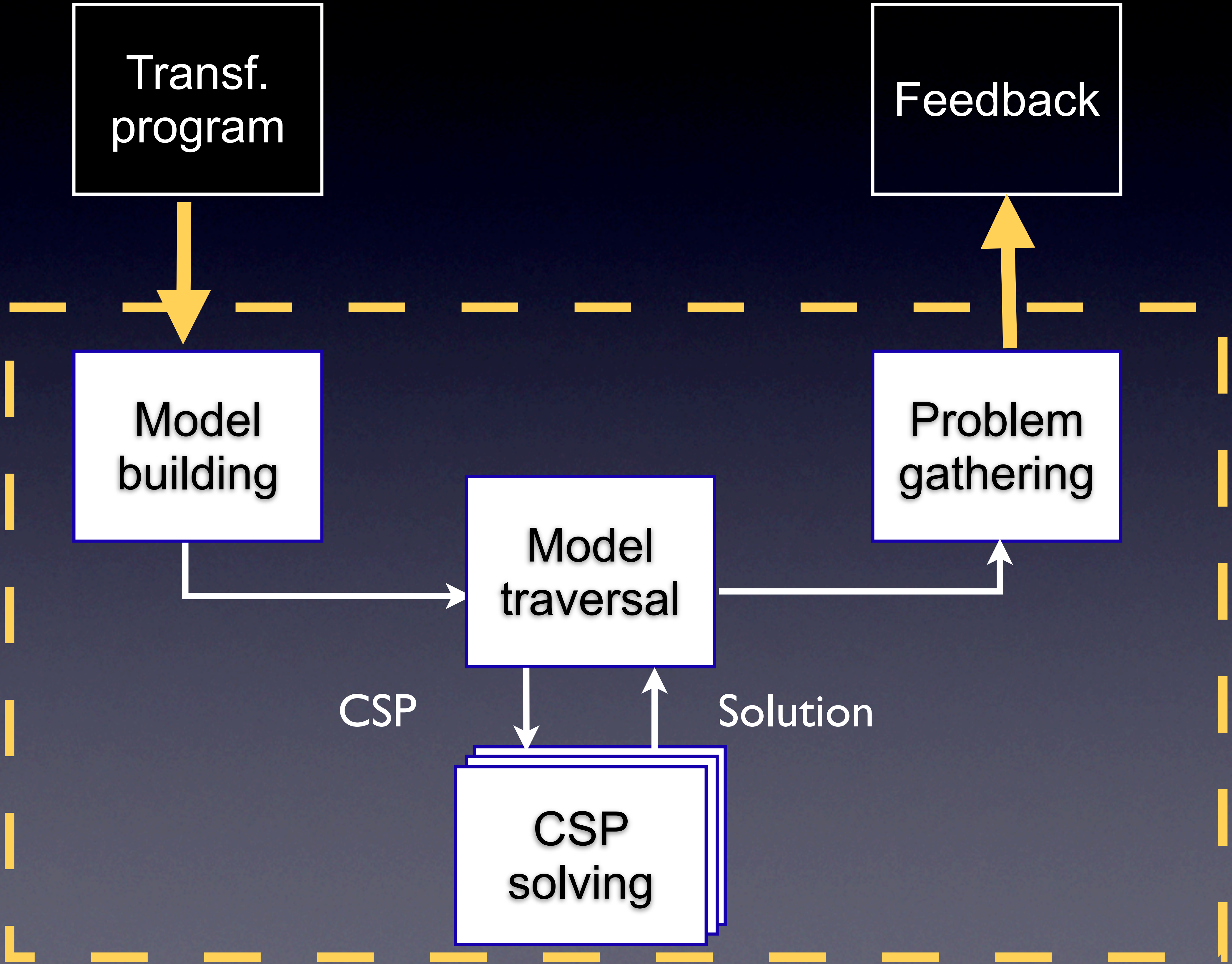
CSP solving

Problem gathering

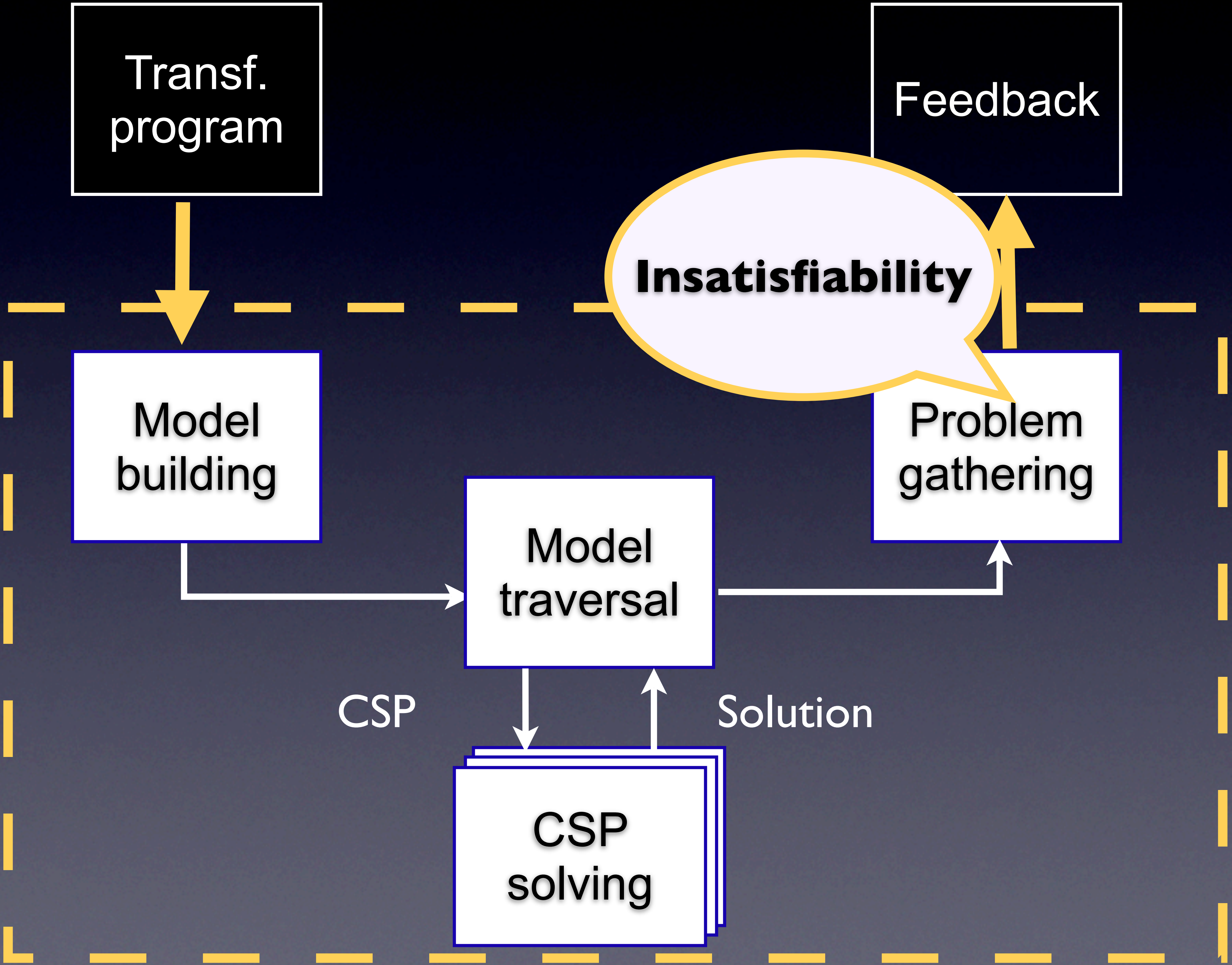
Multiple Solvers

Incremental Evaluation

Solution


Transf. program
Feedback
Model building
Problem gathering
Model traversal
CSP
Solution
CSP solving

Solution



Transf. program

Feedback

Insatisfiability

Model building

Model traversal

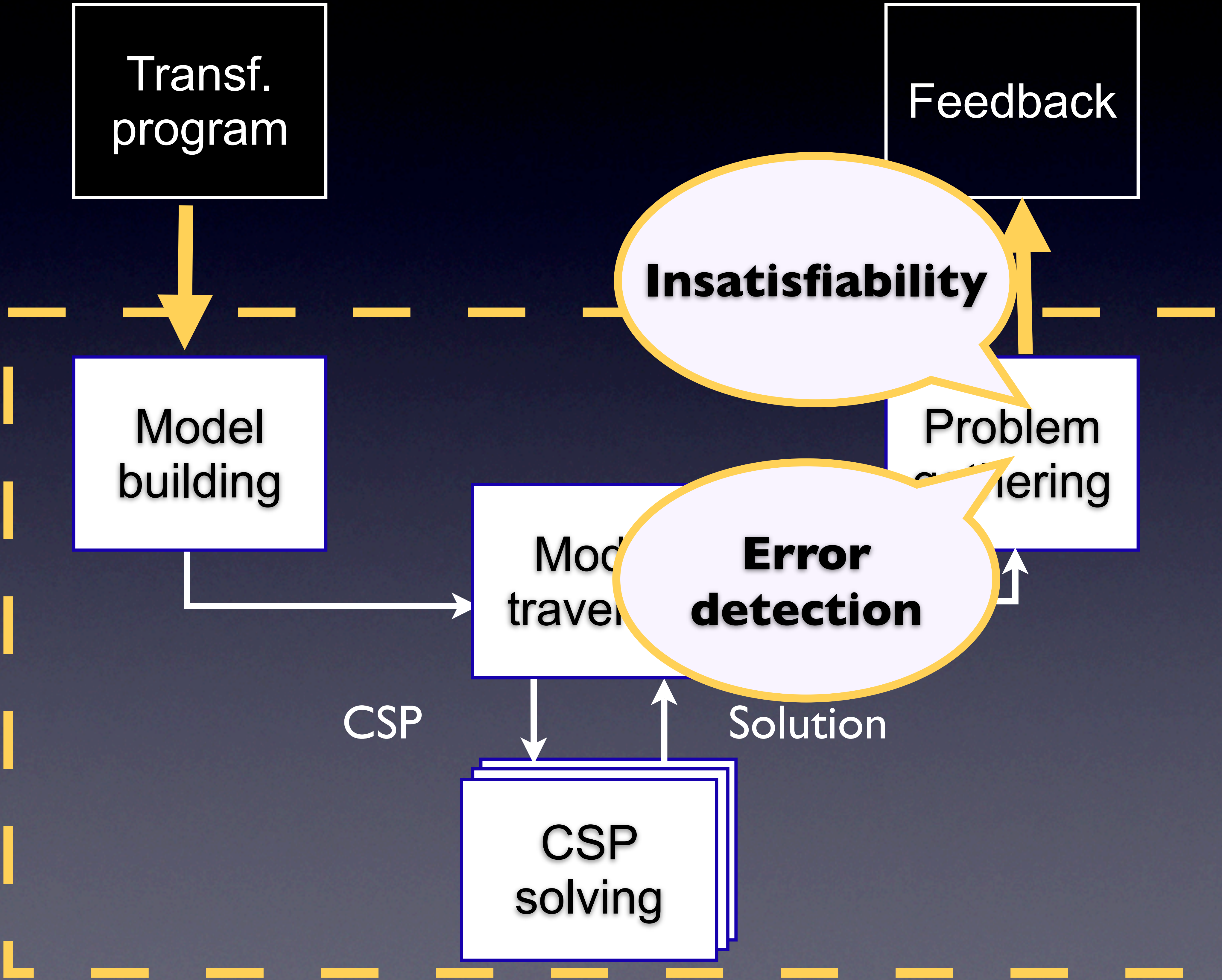
Problem gathering

CSP

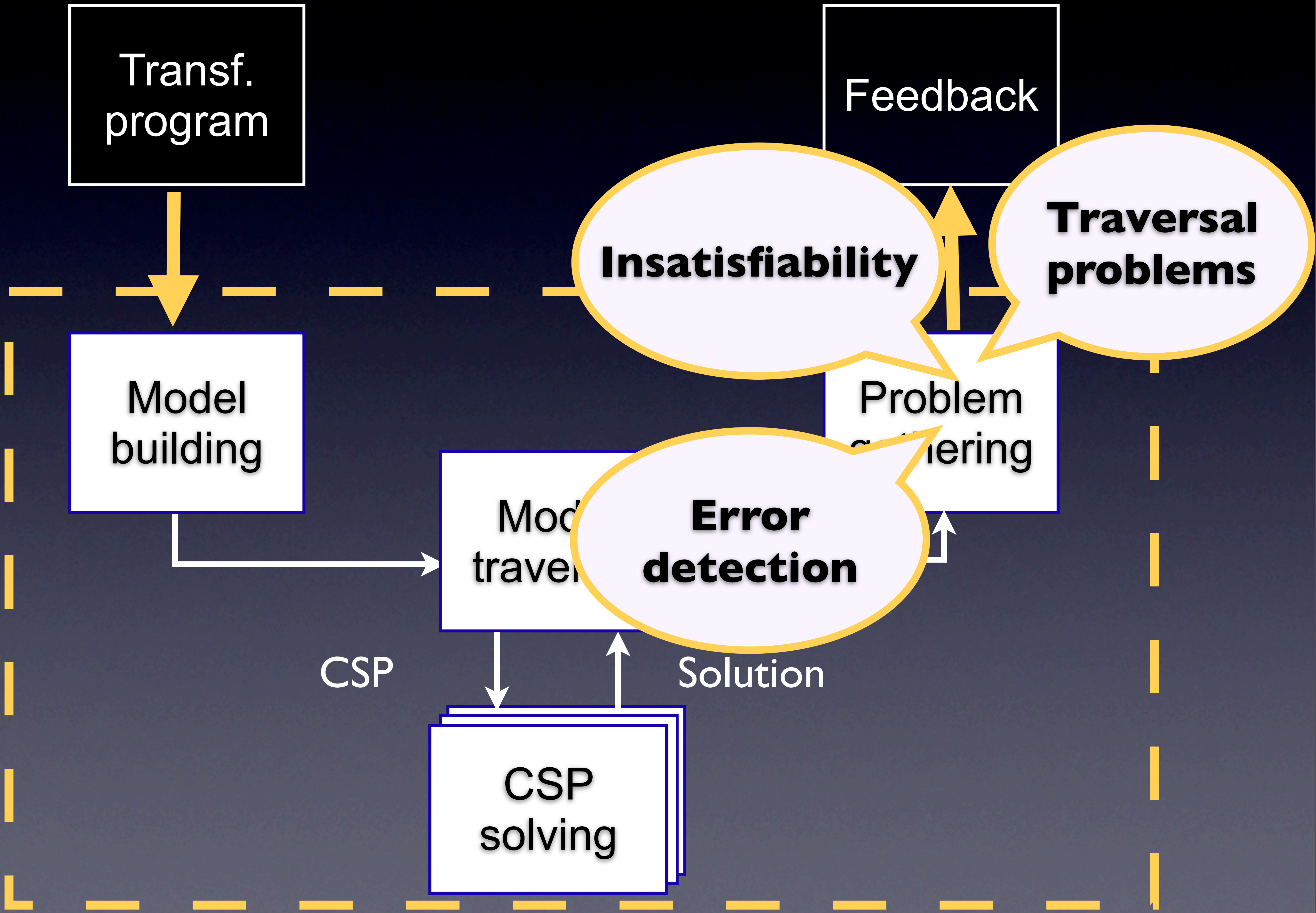
Solution

CSP solving

Solution


Transf. program
Feedback
Insatisfiability
Model building
Problem g…hering
Mod… travel…
Error detection
CSP
Solution
CSP solving

Solution


Transf. program
Feedback
Insatisfiability
Traversal problems
Model building
Problem gathering
Model traversal
Error detection
CSP
Solution
CSP solving

Identified problems

- **Type system errors**
  - Invalid types (insatisfiability)
  - Type changes (detector)
- **Control structure errors**
  - Unhandled failures (traversal)
  - Missing entry point (traversal)
  - Dead code: unused rule/pattern (traversal)

Future Plans

- **Achieved Results**
  - **Type safety ⇒ CSP**
    - Full language support
    - Evaluated multiple CSP solvers
  - **Modular Traversal**
    - Contract handling
- **Future Plans**
  - Analysis of further properties
  - New detectors
  - More precise error identification