Fauré: a Partial Approach to Network Analysis

Fangping Lan, Bin Gui, and Anduo Wang
Temple University

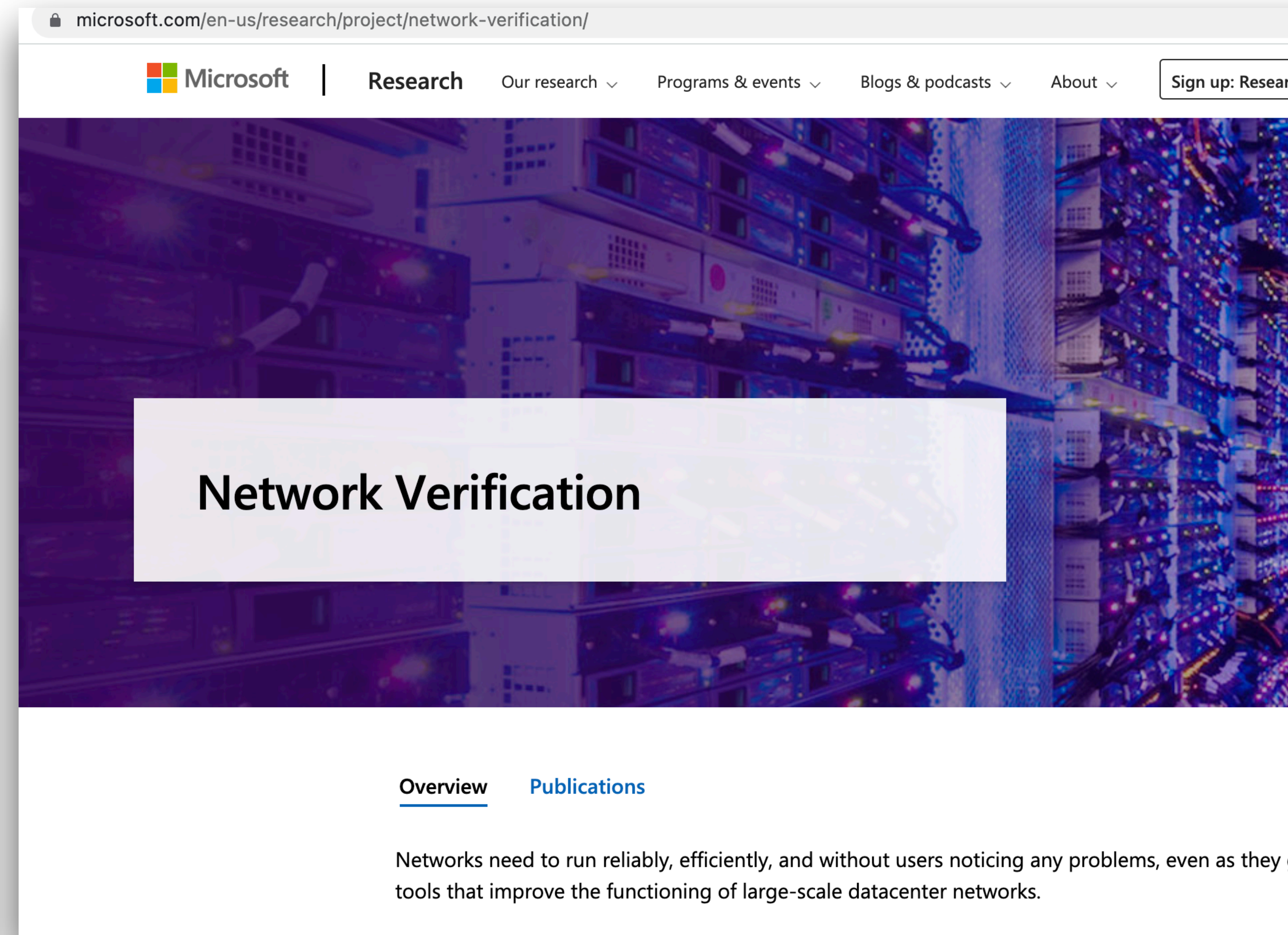
HotNets 2021, Nov 11

network analysis — significant progress

[NSDI 20] Tiramisu: Fast Multilayer Network Verification.
[SIGCOMM'02] Route oscillations in I-BGP with route reflection.
[HotNets'20] Solver-Aided Multi-Party Configuration.
[NSDI'15] General Approach to Network Configuration Analysis.
[SIGCOMM'16] Fast Control Plane Analysis Using an Abstract Representation.
[TON'02] The Stable Paths Problem and Interdomain Routing.
[SIGCOMM'19] Validating Datacenters at Scale.
[CoNEXT 20] AalWiNes: A Fast and Quantitative What-If Analysis Tool for MPLS Networks.
[NSDI 13] Real Time Network Policy Checking Using Header Space Analysis
[HotSDN 12] VeriFlow: Verifying Network-Wide Invariants in Real Time
[NSDI 15] Checking Beliefs in Dynamic Net- works.
[POPL 16] Scaling Network Verification Using Symmetry and Surgery
[NSDI 20] Plankton: Scalable network config- uration verification through model checking
[IEEE Networks 05] Modeling the routing of an autonomous system with C-BGP.
[INFOCOM 18] Polynomial-Time What-If Analysis for Prefix-Manipulating MPLS Networks
[SIGCOMM 19] Safely and Automatically Updating In-Network ACL Configurations with Intent Language.
[INFOCOM 05] On static reachability analysis of IP networks
[SIGCOMM 20] Accuracy, Scalability, Coverage: A Practical Configuration Verifier on a Global WAN
[HotNets 20] Incremental Network Configuration Verification
[NSDI 20] APKeep: Realtime Verification for Real Networks
… …

network analysis — significant progress

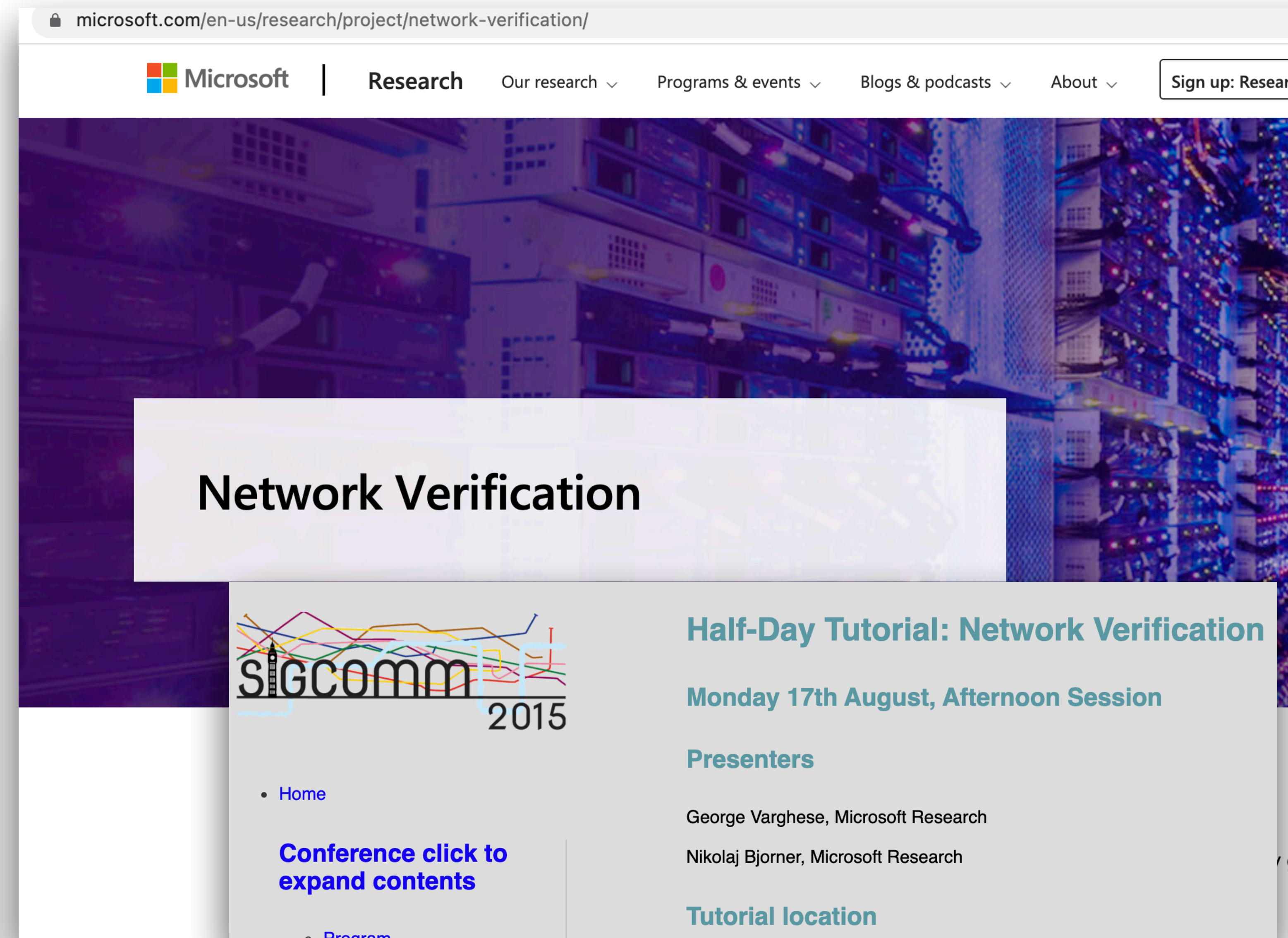
[NSDI 20] Tiramisu: Fast Multilayer Network Verification.
[SIGCOMM’02] Route oscillations in I-BGP with route reflection.
[HotNets’20] Solver-Aided Multi-Party Configuration.
[NSDI’15] General Approach to Network Configuration Analysis.
[SIGCOMM’16] Fast Control Plane Analysis Using an Abstract Representation.
[TON’02] The Stable Paths Problem and Interdomain Routing.
[SIGCOMM’19] Validating Datacenters at Scale.
[CoNEXT 20] AalWiNes: A Fast and Quantitative What-If Analysis Tool for MPLS Networks.
[NSDI 13] Real Time Network Policy Checking Using Header Space Analysis
[HotSDN 12] VeriFlow: Verifying Network-Wide Invariants in Real Time
[NSDI 15] Checking Beliefs in Dynamic Net- works.
[POPL 16] Scaling Network Verification Using Symmetry and Surgery
[NSDI 20] Plankton: Scalable network config- uration verification through model checking
[IEEE Networks 05] Modeling the routing of an autonomous system with C-BGP.
[INFOCOM 18] Polynomial-Time What-If Analysis for Prefix-Manipulating MPLS Networks
[SIGCOMM 19] Safely and Automatically Updating In-Network ACL Configurations with Intent Language.
[INFOCOM 05] On static reachability analysis of IP networks
[SIGCOMM 20] Accuracy, Scalability, Coverage: A Practical Configuration Verifier on a Global WAN
[HotNets 20] Incremental Network Configuration Verification
[NSDI 20] APKeep: Realtime Verification for Real Networks
… …



network analysis — significant progress

[NSDI 20] Tiramisu: Fast Multilayer Network Verification.
[SIGCOMM'02] Route oscillations in I-BGP route reflection.
[HotNets'20] Solver-Aided Multi-Party Configuration.
[NSDI'15] General Approach to Network Configuration Analysis.
[SIGCOMM'16] Fast Control Plane Analysis Using an Abstract Representation.
[TON'02] The Stable Paths Problem and Interdomain Routing.
[SIGCOMM'19] Validating Datacenters at Scale.
[CoNEXT 20] AalWiNes: A Fast and Quantitative What-If Analysis Tool for MPLS Networks.
[NSDI 13] Real Time Network Policy Checking Using Header Space Analysis
[HotSDN 12] VeriFlow: Verifying Network-Wide Invariants in Real Time
[NSDI 15] Checking Beliefs in Dynamic Net- works.
[POPL 16] Scaling Network Verification Using Symmetry and Surgery
[NSDI 20] Plankton: Scalable network config- uration verification through model checking
[IEEE Networks 05] Modeling the routing of an autonomous system with C-BGP.
[INFOCOM 18] Polynomial-Time What-If Analysis for Prefix-Manipulating MPLS Networks
[SIGCOMM 19] Safely and Automatically Updating In-Network ACL Configurations with Intent Language.
[INFOCOM 05] On static reachability analysis of IP networks
[SIGCOMM 20] Accuracy, Scalability, Coverage: A Practical Configuration Verifier on a Global WAN
[HotNets 20] Incremental Network Configuration Verification
[NSDI 20] APKeep: Realtime Verification for Real Networks
… …

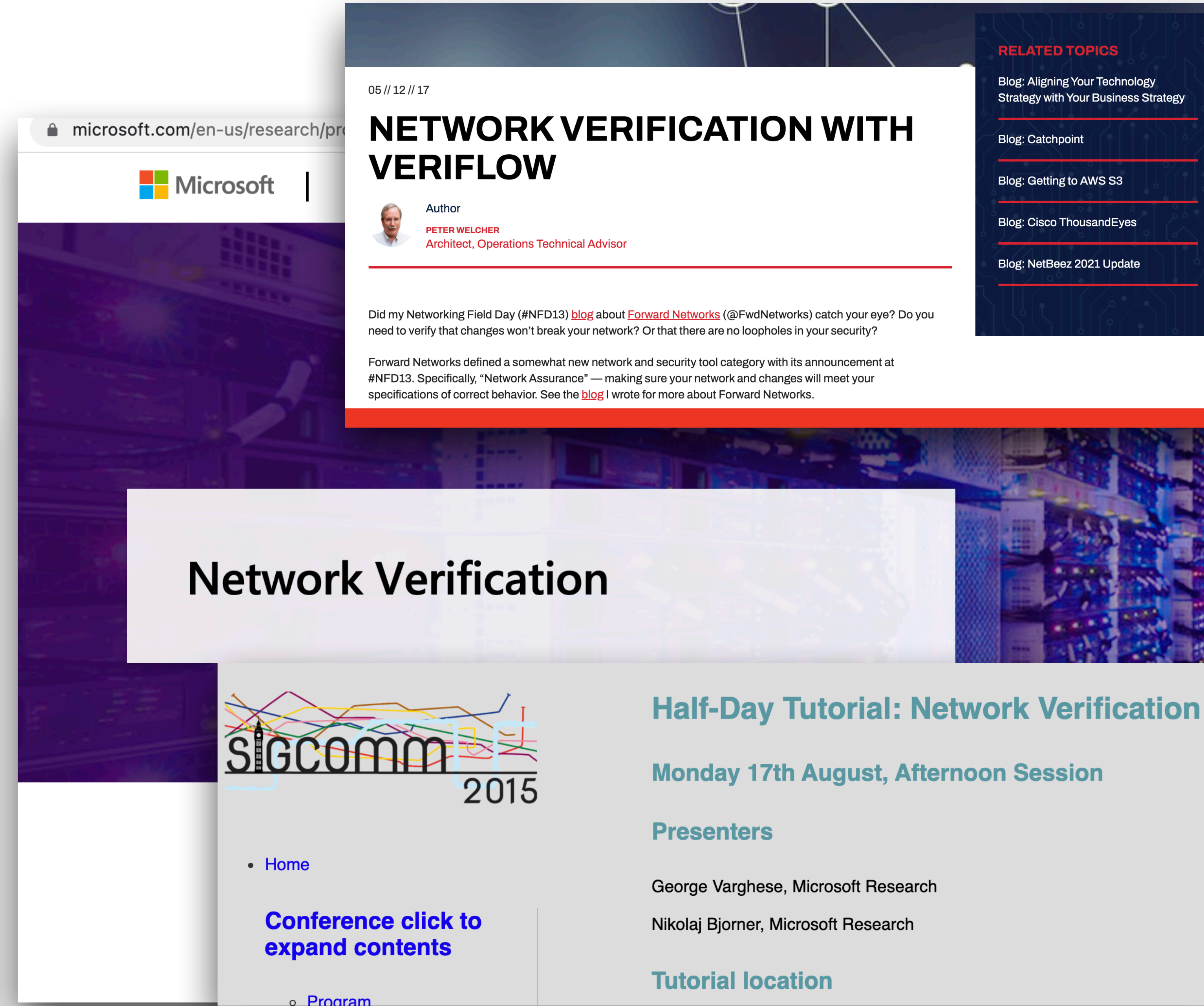
with



network analysis — significant progress

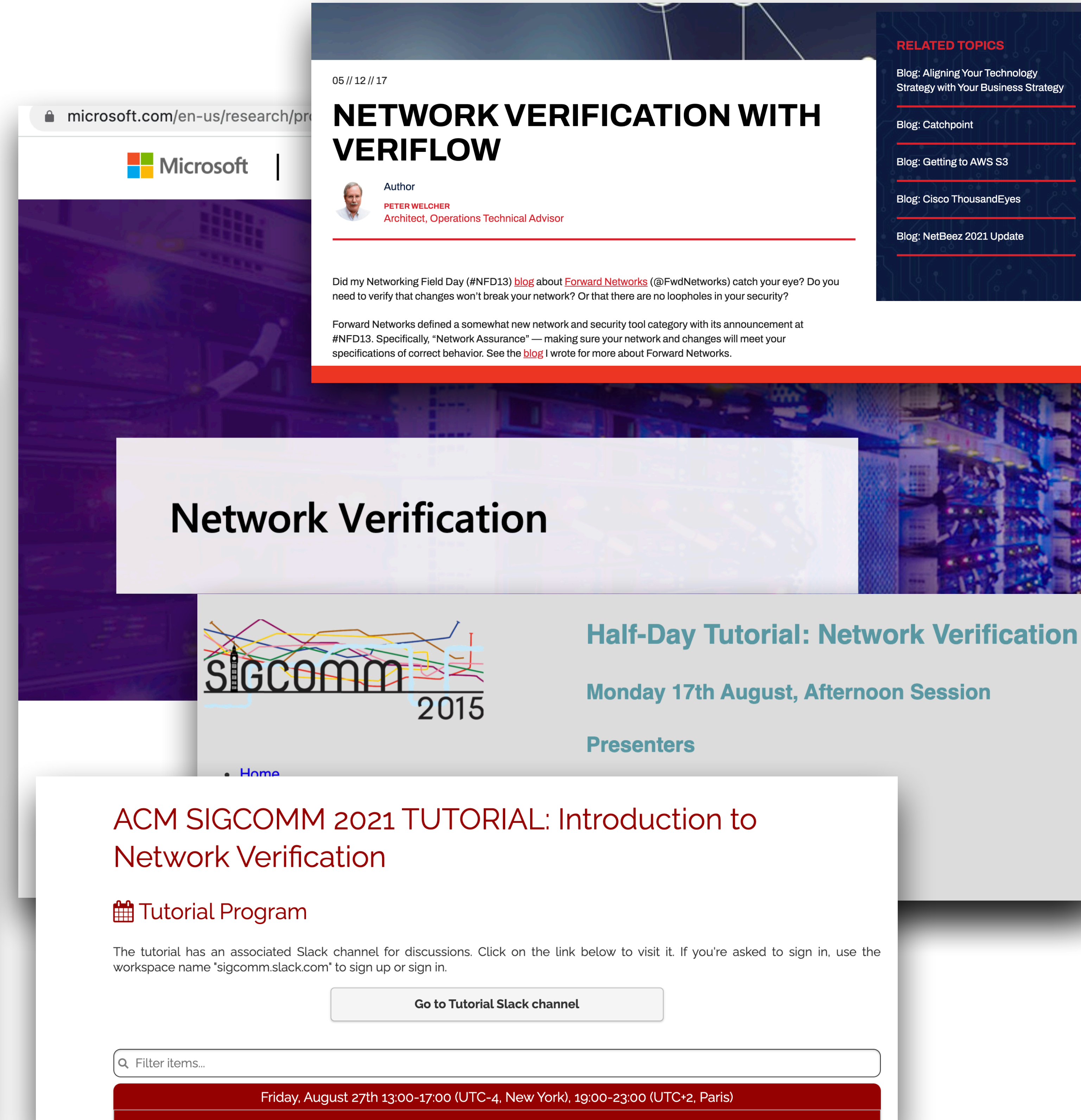
[NSDI 20] Tiramisu: Fast Multilayer Network Verification.
[SIGCOMM’02] Route oscillations in I-BGP route reflection.
[HotNets’20] Solver-Aided Multi-Party Configuration.
[NSDI’15] General Approach to Network Configuration Analysis.
[SIGCOMM’16] Fast Control Plane Analysis Using an Abstract Representation.
[TON’02] The Stable Paths Problem and Interdomain Routing.
[SIGCOMM’19] Validating Datacenters at Scale.
[CoNEXT 20] AalWiNes: A Fast and Quantitative What-If Analysis Tool for MPLS Networks.
[NSDI 13] Real Time Network Policy Checking Using Header Space Analysis
[HotSDN 12] VeriFlow: Verifying Network-Wide Invariants in Real Time
[NSDI 15] Checking Beliefs in Dynamic Net- works.
[POPL 16] Scaling Network Verification Using Symmetry and Surgery
[NSDI 20] Plankton: Scalable network config- uration verification through model checking
[IEEE Networks 05] Modeling the routing of an autonomous system with C-BGP.
[INFOCOM 18] Polynomial-Time What-If Analysis for Prefix-Manipulating MPLS Networks
[SIGCOMM 19] Safely and Automatically Updating In-Network ACL Configurations with Intent Language.
[INFOCOM 05] On static reachability analysis of IP networks
[SIGCOMM 20] Accuracy, Scalability, Coverage: A Practical Configuration Verifier on a Global WAN
[HotNets 20] Incremental Network Configuration Verification
[NSDI 20] APKeep: Realtime Verification for Real Networks
… …

with

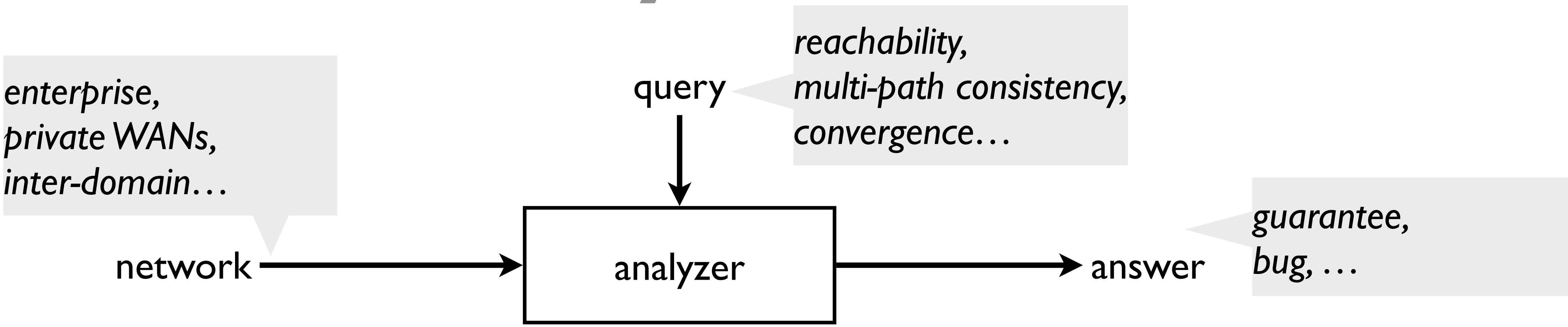


network analysis — significant progress

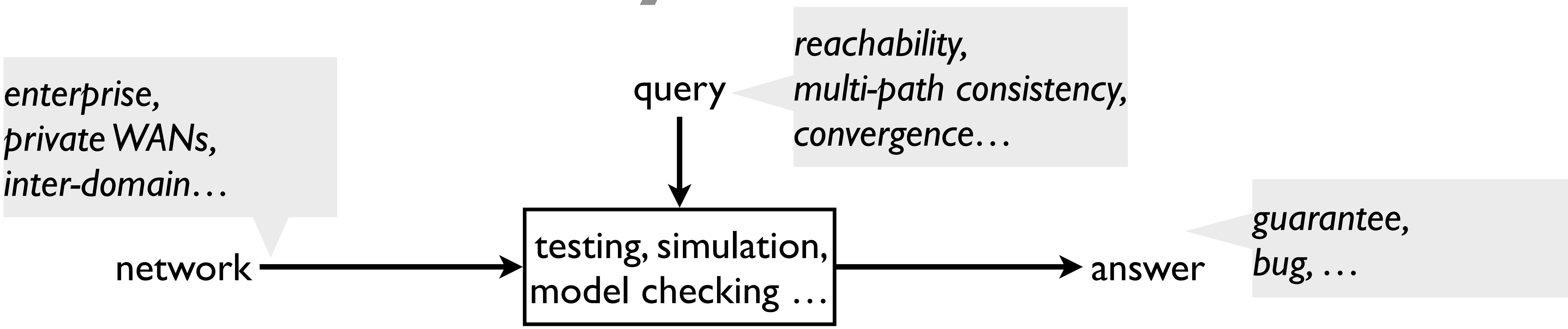
[NSDI 20] Tiramisu: Fast Multilayer Network Verification.
[SIGCOMM'02] Route oscillations in I-BGP with route reflection.
[HotNets'20] Solver-Aided Multi-Party Configuration.
[NSDI'15] General Approach to Network Configuration Analysis.
[SIGCOMM'16] Fast Control Plane Analysis Using an Abstract Representation.
[TON'02] The Stable Paths Problem and Interdomain Routing.
[SIGCOMM'19] Validating Datacenters at Scale.
[CoNEXT 20] AalWiNes: A Fast and Quantitative What-If Analysis Tool for MPLS Networks.
[NSDI 13] Real Time Network Policy Checking Using Header Space Analysis
[HotSDN 12] VeriFlow: Verifying Network-Wide Invariants in Real Time
[NSDI 15] Checking Beliefs in Dynamic Net- works.
[POPL 16] Scaling Network Verification Using Symmetry and Surgery
[NSDI 20] Plankton: Scalable network config- uration verification through model checking
[IEEE Networks 05] Modeling the routing of an autonomous system with C-BGP.
[INFOCOM 18] Polynomial-Time What-If Analysis for Prefix-Manipulating MPLS Networks
[SIGCOMM 19] Safely and Automatically Updating In-Network ACL Configurations with Intent Language.
[INFOCOM 05] On static reachability analysis of IP networks
[SIGCOMM 20] Accuracy, Scalability, Coverage: A Practical Configuration Verifier on a Global WAN
[HotNets 20] Incremental Network Configuration Verification
[NSDI 20] APKeep: Realtime Verification for Real Networks
… …



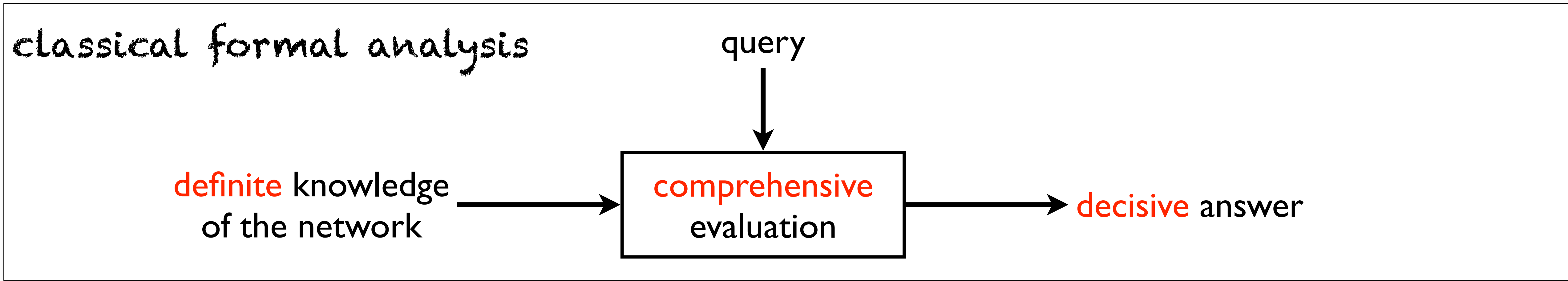
network analysis



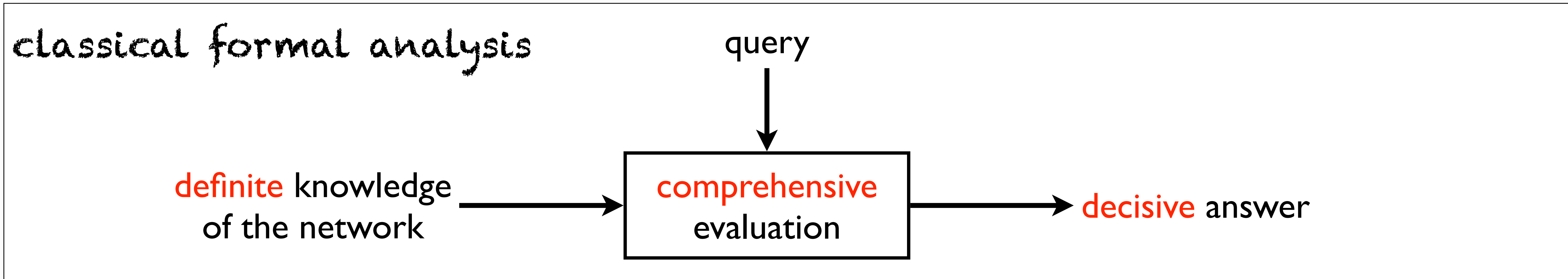
network analysis

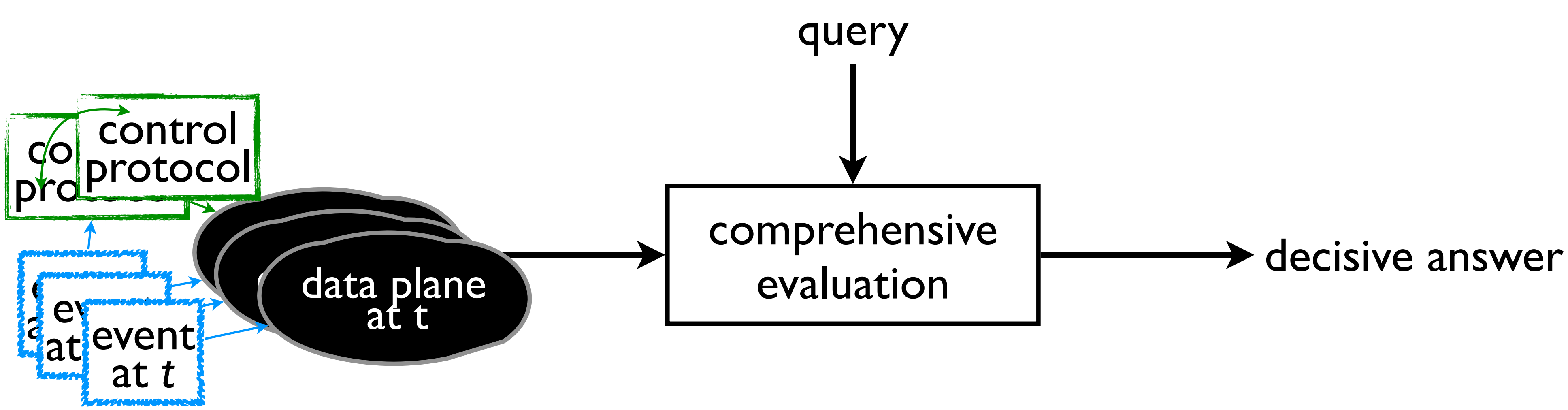


complete network analysis

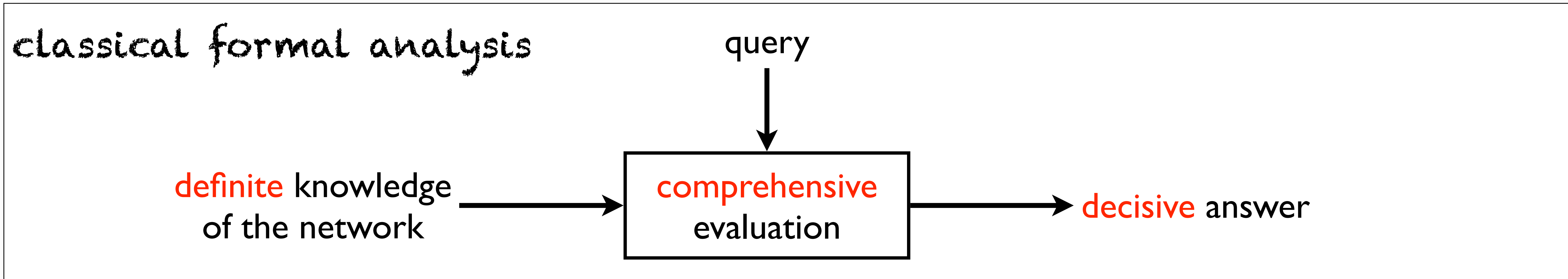


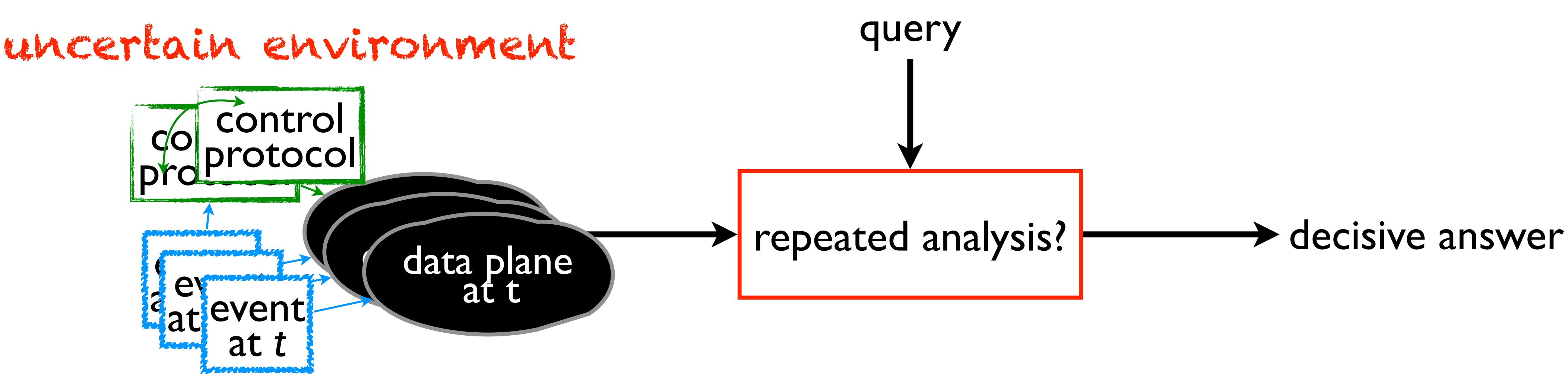
challenges with complete analysis



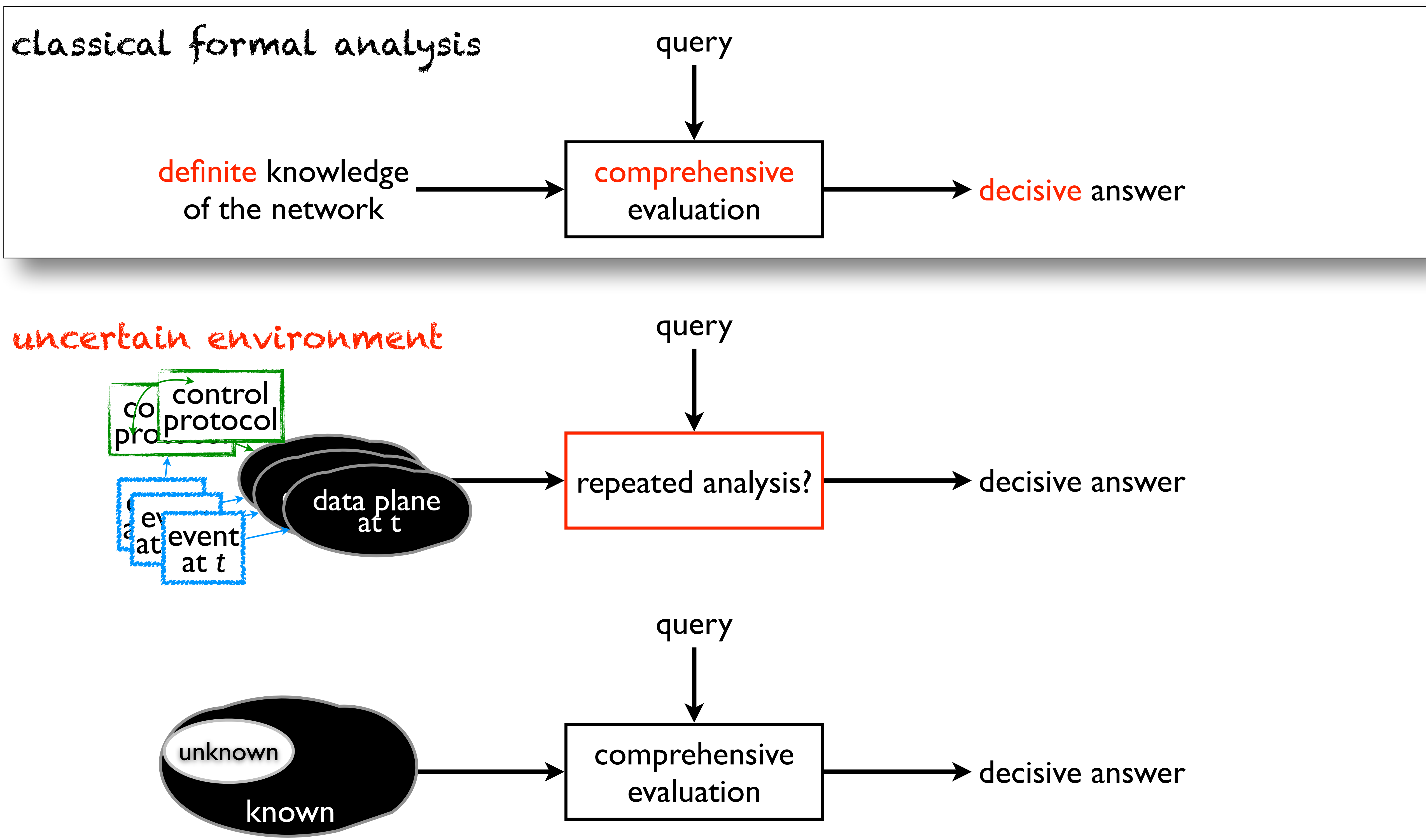


challenges with complete analysis

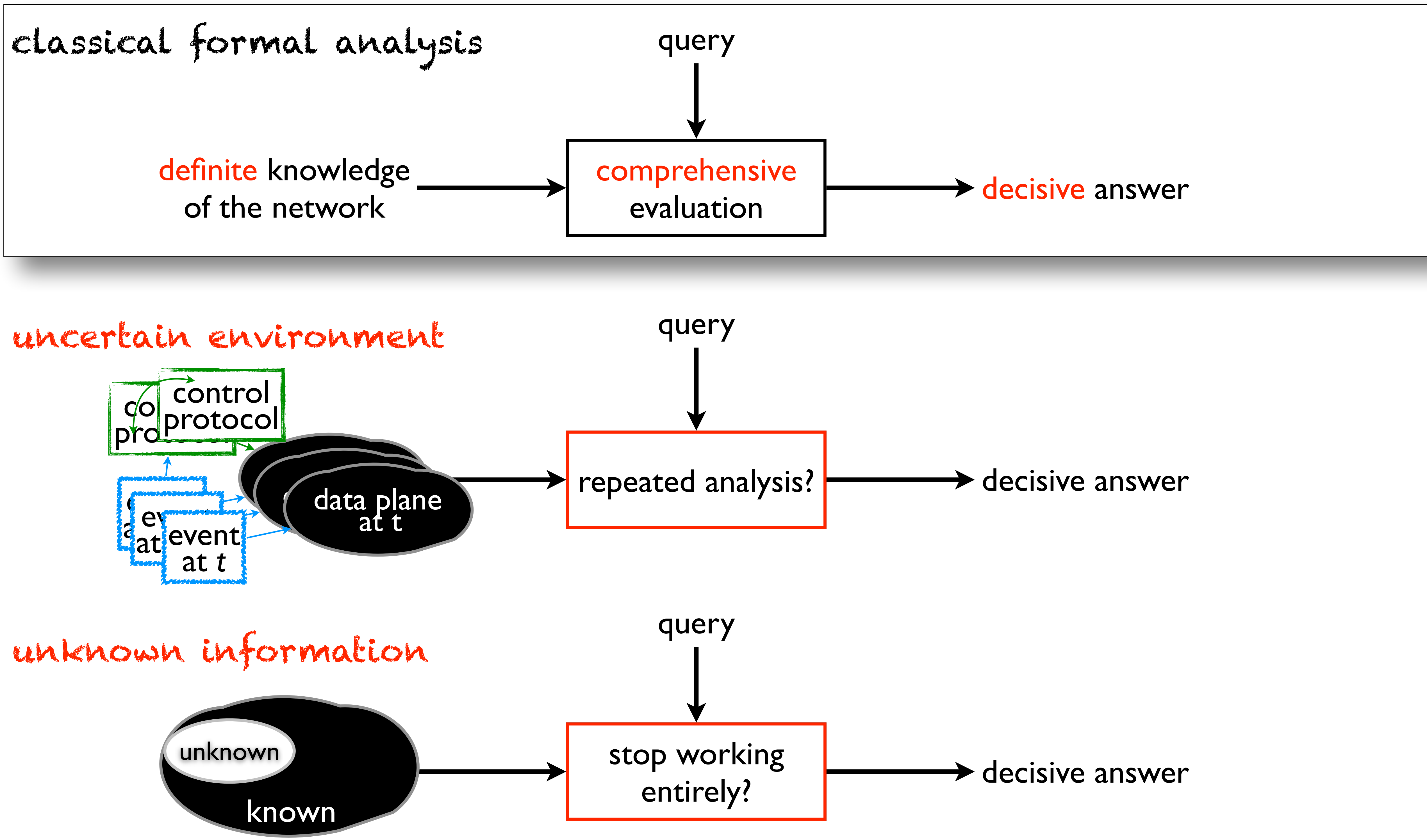




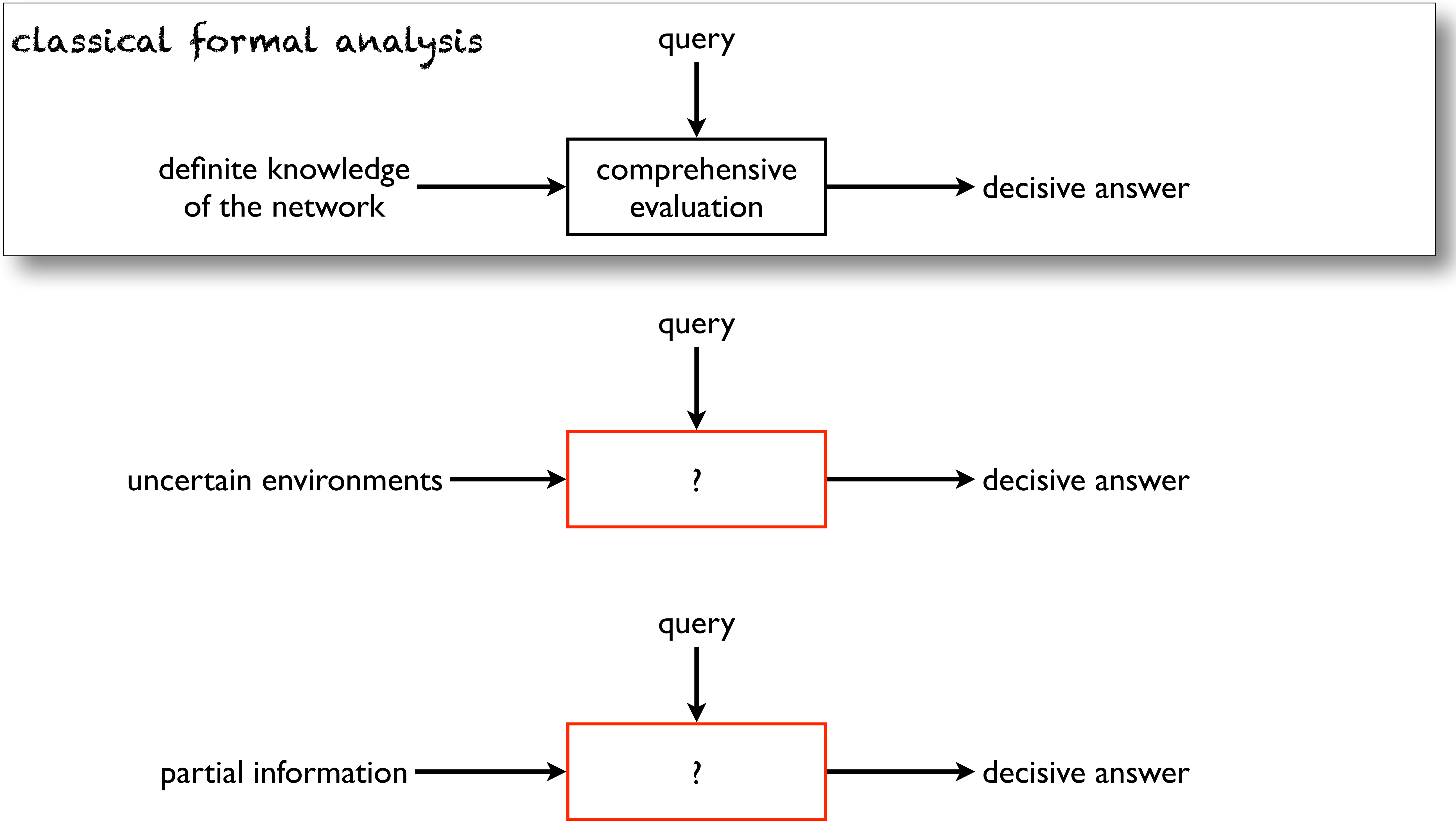
challenges with complete analysis



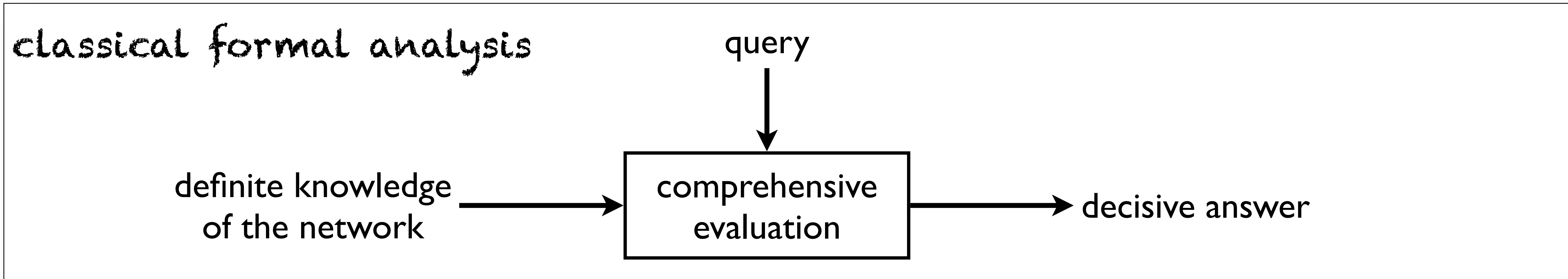
challenges with complete analysis

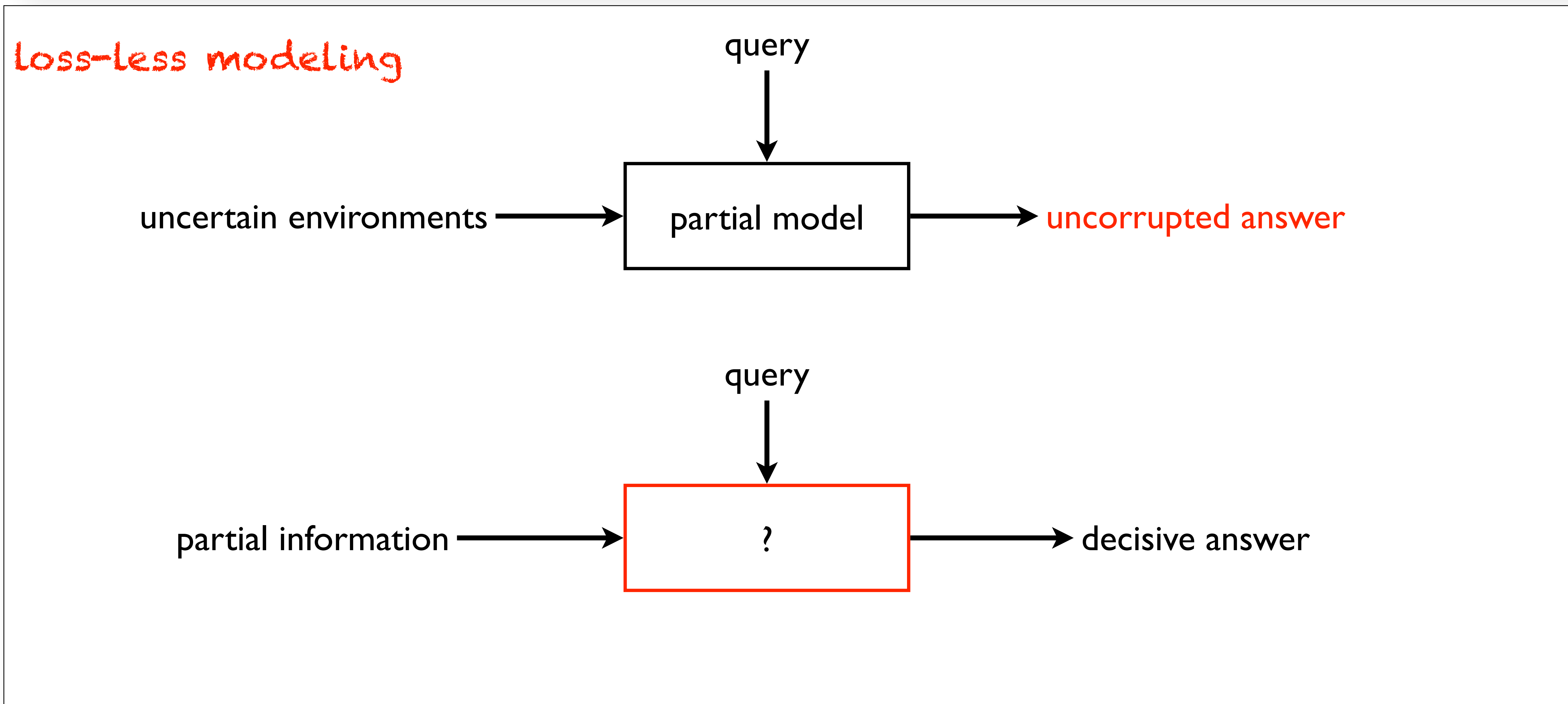


challenges with complete analysis

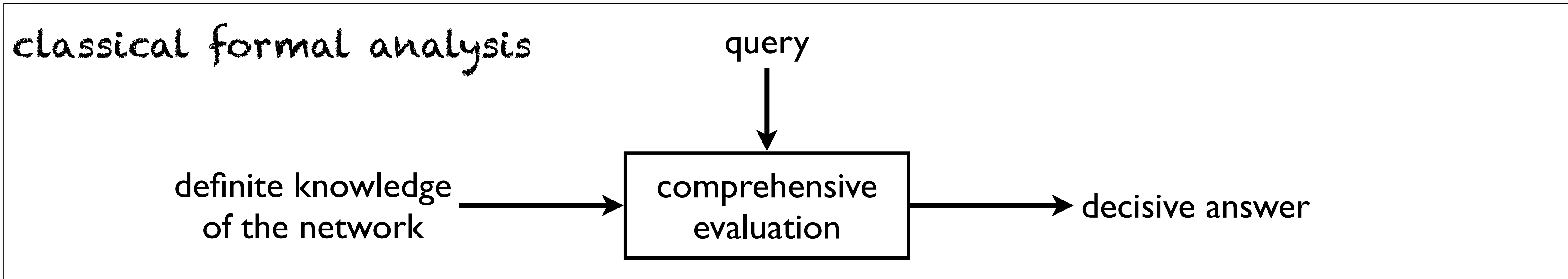


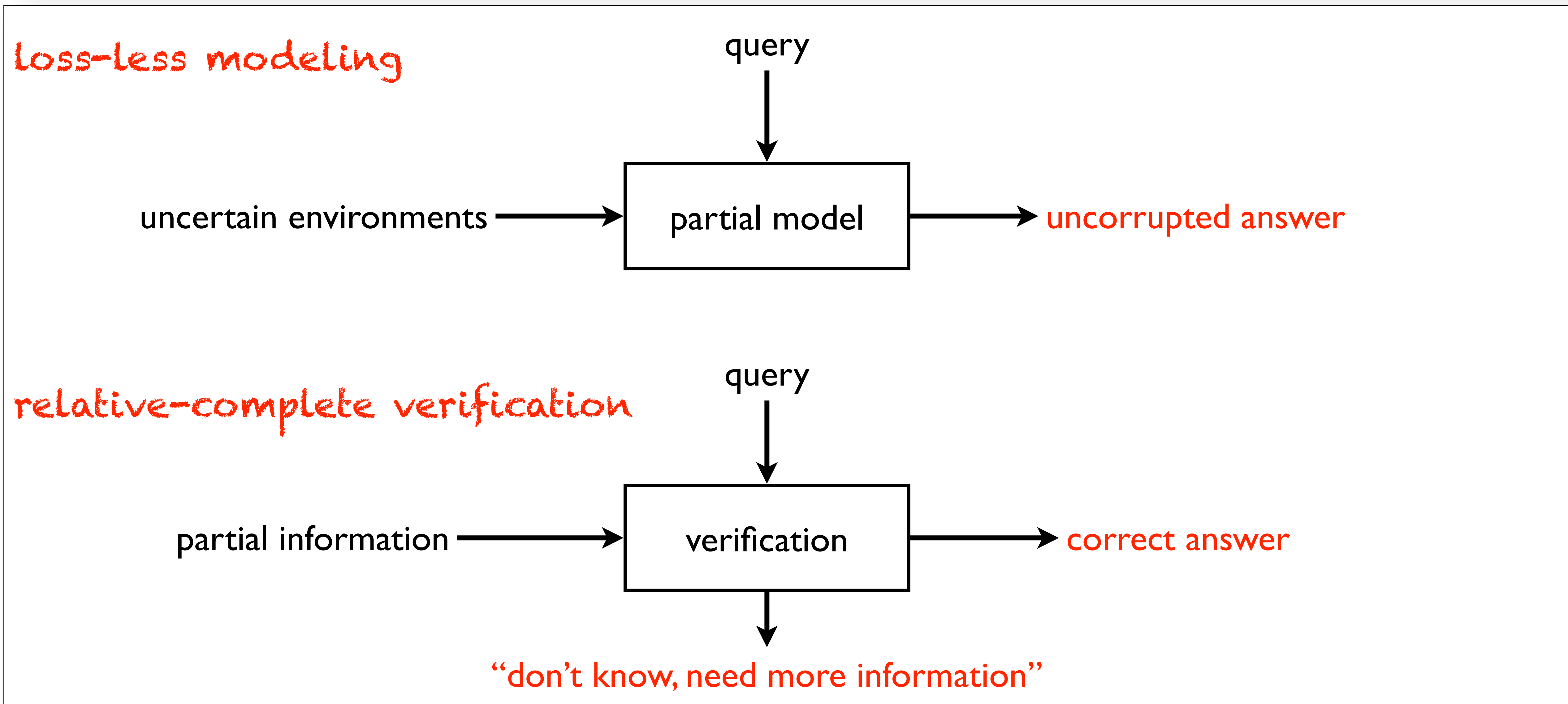
a partial approach



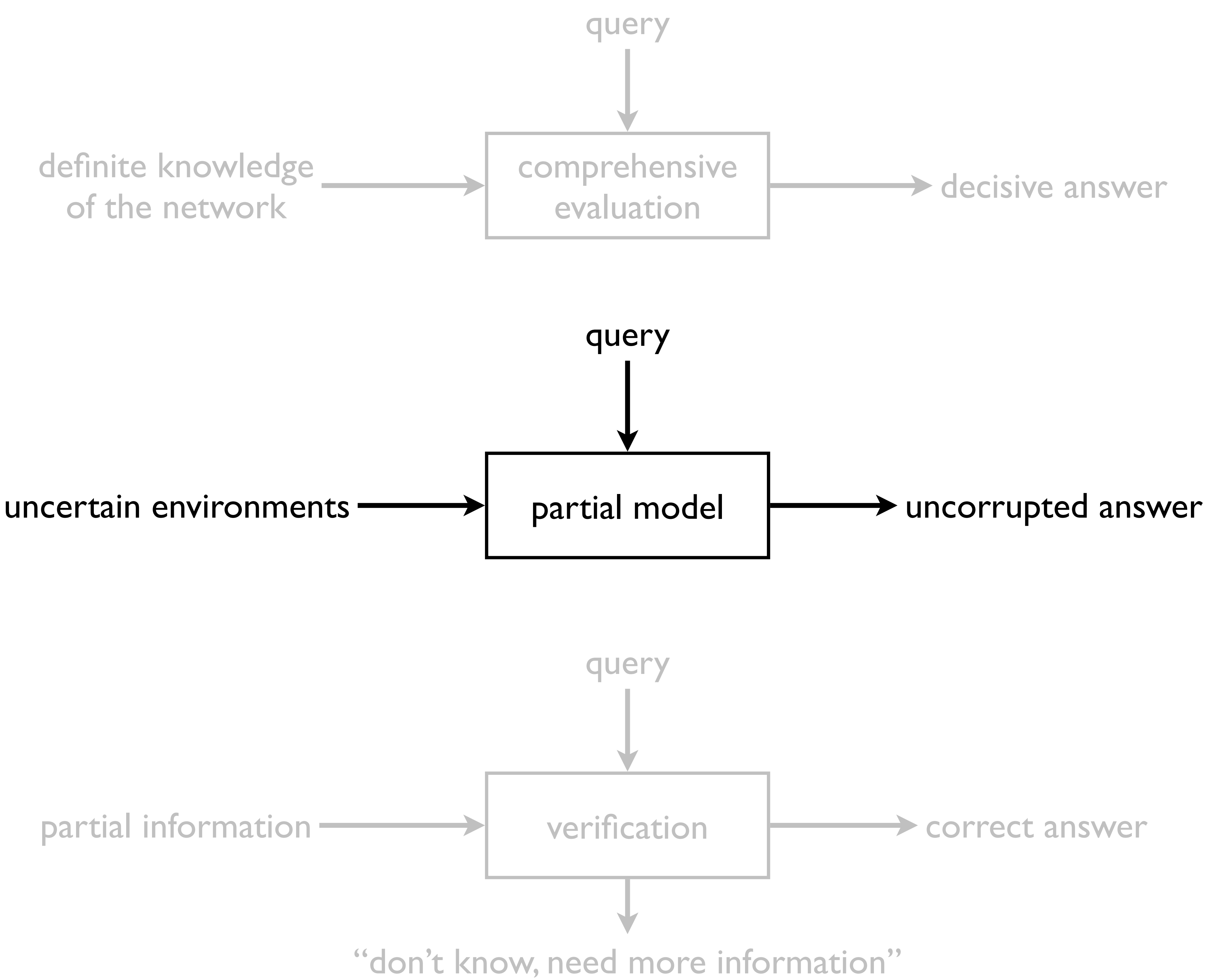


a partial approach

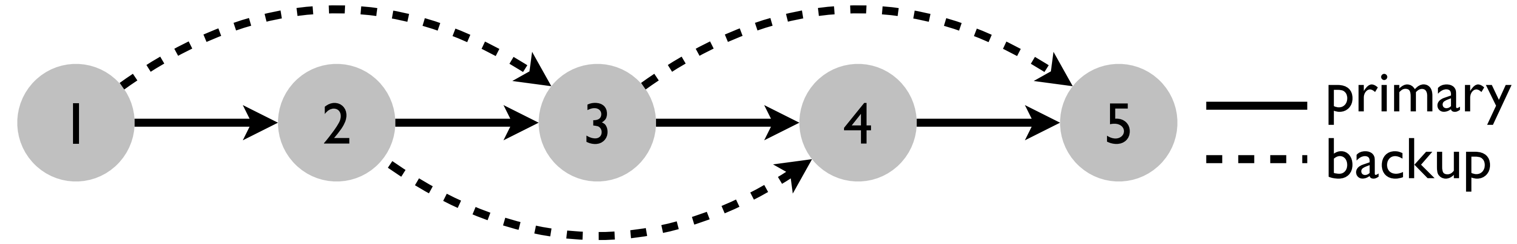


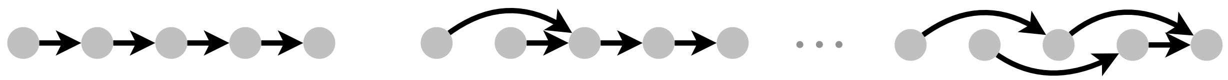


loss-less modeling



modeling





F	node	node
	1	2
	2	3
	3	4
	4	5

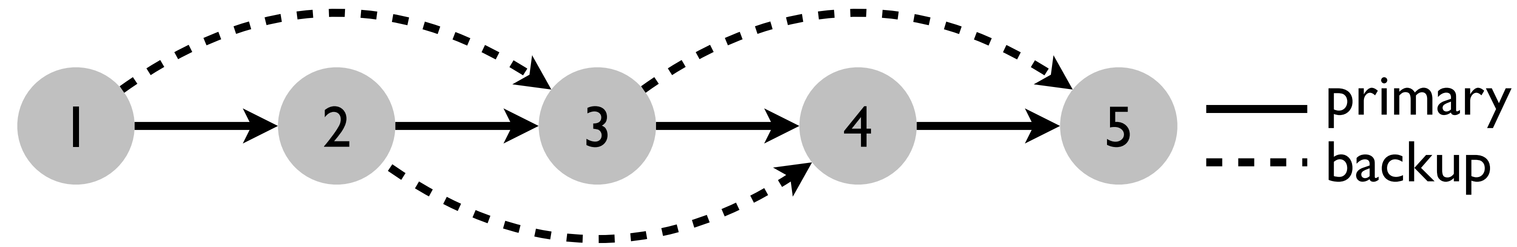
F	node	node
	1	3
	2	3
	3	4
	4	5

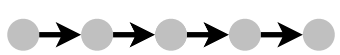
…

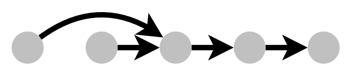
F	node	node
	1	3
	2	4
	3	5
	4	5

forwarding tables

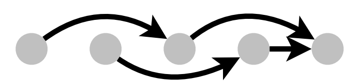
modeling







…



F	node	node
	1	2
	2	3
	3	4
	4	5

F	node	node
	1	3
	2	3
	3	4
	4	5

…

F	node	node
	1	3
	2	4
	3	5
	4	5

forwarding tables

(reachability) query

R	src	dest
	1	2
	1	3
	1	4
	1	5
	2	3
	2	4

…

R	src	dest
	1	3
	1	4
	1	5
	2	3
	2	4

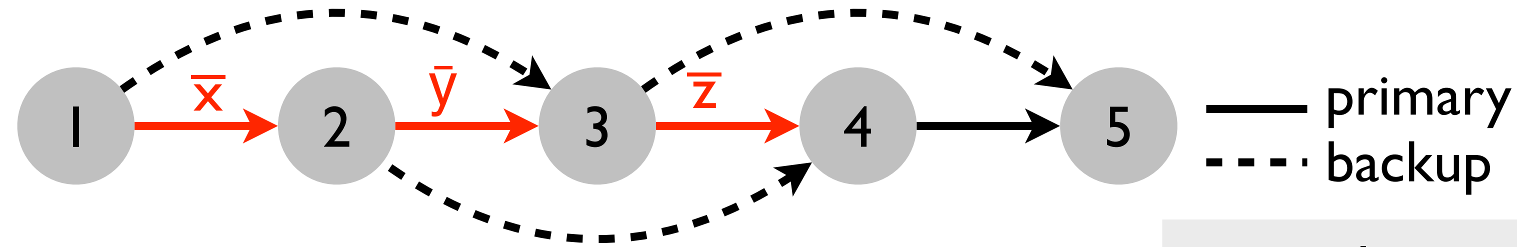
…

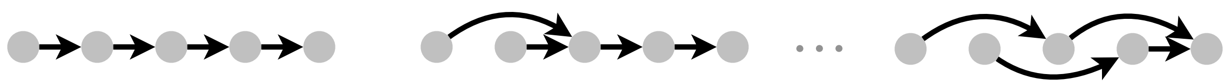
…

R	src	dest
	1	3
	1	5
	2	4

…

modeling





F	node	node
	1	2
	2	3
	3	4
	4	5

F	node	node
	1	3
	2	3
	3	4
	4	5

…

F	node	node
	1	3
	2	4
	3	5
	4	5

Rep

c-table:

F	node	node	
	1	2	$\bar{x}=1$
	1	3	$\bar{x}=0$
	2	3	$\bar{y}=1$
	2	4	$\bar{y}=0$

…

a tuple can occur only when the condition is satisfied

1: normal
0: failed

query

R	src	dest
	1	2
	1	3
	1	4
	1	5
	2	3
	2	4

…

R	src	dest
	1	3
	1	4
	1	5
	2	3
	2	4

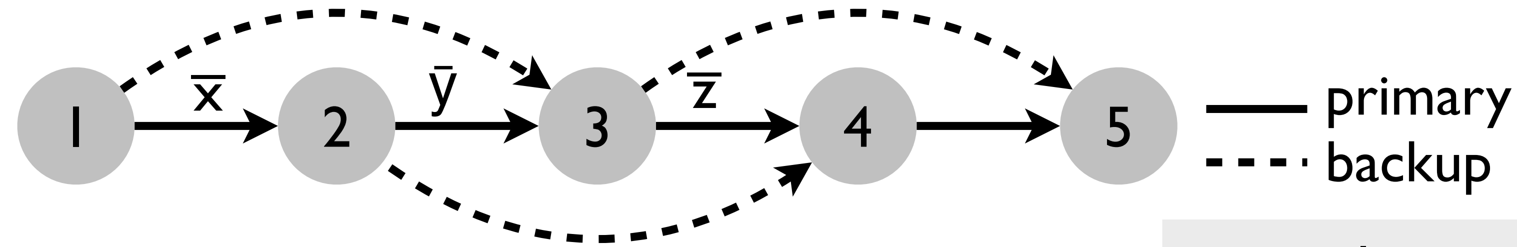
…

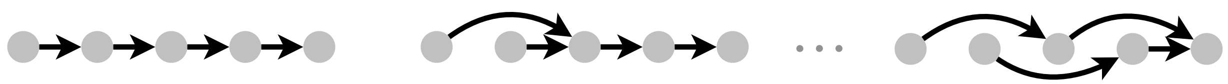
…

R	src	dest
	1	3
	1	5
	2	4

…

modeling





F	node	node
	1	2
	2	3
	3	4
	4	5

F	node	node
	1	3
	2	3
	3	4
	4	5

…

F	node	node
	1	3
	2	4
	3	5
	4	5

Rep

F	node	node	
	1	2	$\bar{x}=1$
	1	3	$\bar{x}=0$
	2	3	$\bar{y}=1$
	2	4	$\bar{y}=0$

…

a tuple can occur only when the condition is satisfied

1: normal
0: failed

query

query

R	src	dest
	1	2
	1	3
	1	4
	1	5
	2	3
	2	4

…

R	src	dest
	1	3
	1	4
	1	5
	2	3
	2	4

…

…

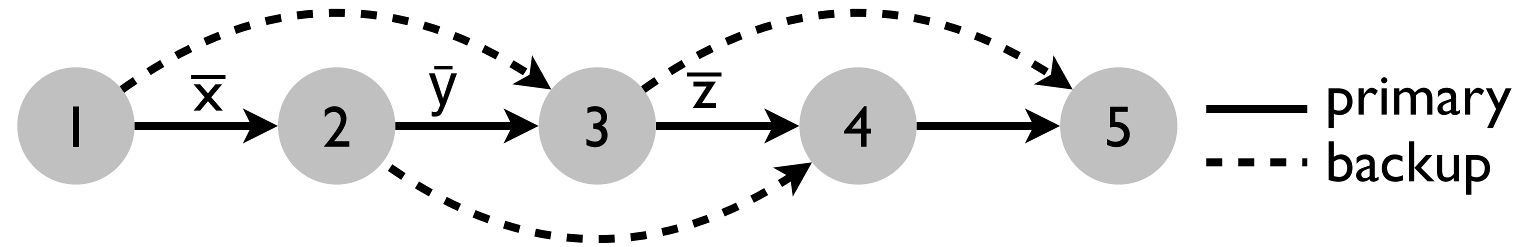
R	src	dest
	1	3
	1	5
	2	4

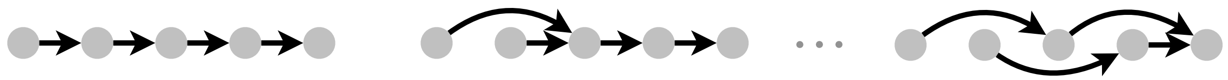
…

R	src	dest	
	1	2	$\bar{x}=1$
	…		
	1	5	$\bar{x}=1 \wedge \bar{y}=1 \wedge \bar{z}=1$
	1	5	$\bar{x}=0 \wedge \bar{z}=1$
	1	5	$\bar{x}=0 \wedge \bar{z}=0$
	1	5	$\bar{x}=1 \wedge \bar{y}=0$
	2	3	$\bar{y}=1$

…

modeling





F	node	node
	1	2
	2	3
	3	4
	4	5

F	node	node
	1	3
	2	3
	3	4
	4	5

…

F	node	node
	1	3
	2	4
	3	5
	4	5

Rep →

F	node	node	
	1	2	$\bar{x}=1$
	1	3	$\bar{x}=0$
	2	3	$\bar{y}=1$
	2	4	$\bar{y}=0$

…

↓ query

R	src	dest
	1	2
	1	3
	1	4
	1	5
	2	3
	2	4

…

R	src	dest
	1	3
	1	4
	1	5
	2	3
	2	4

…

R	src	dest
	1	3
	1	5
	2	4

…

Rep →

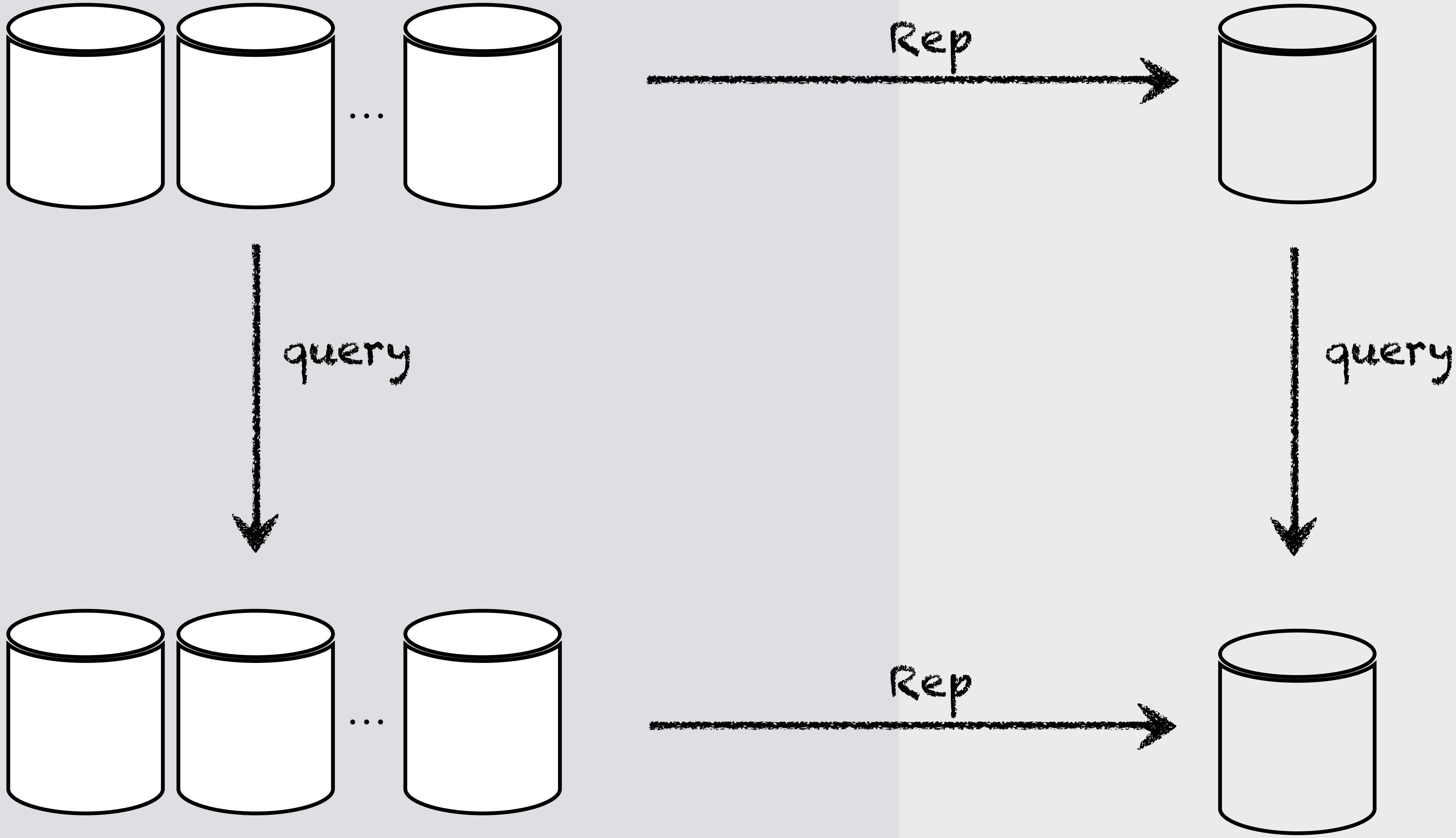
R	src	dest	
	1	2	$\bar{x}=1$
	…		
	1	5	$\bar{x}=1 \wedge \bar{y}=1 \wedge \bar{z}=1$
	1	5	$\bar{x}=0 \wedge \bar{z}=1$
	1	5	$\bar{x}=0 \wedge \bar{z}=0$
	1	5	$\bar{x}=1 \wedge \bar{y}=0$
	2	3	$\bar{y}=1$

…

loss-less modeling

all definite instances (regular tables)

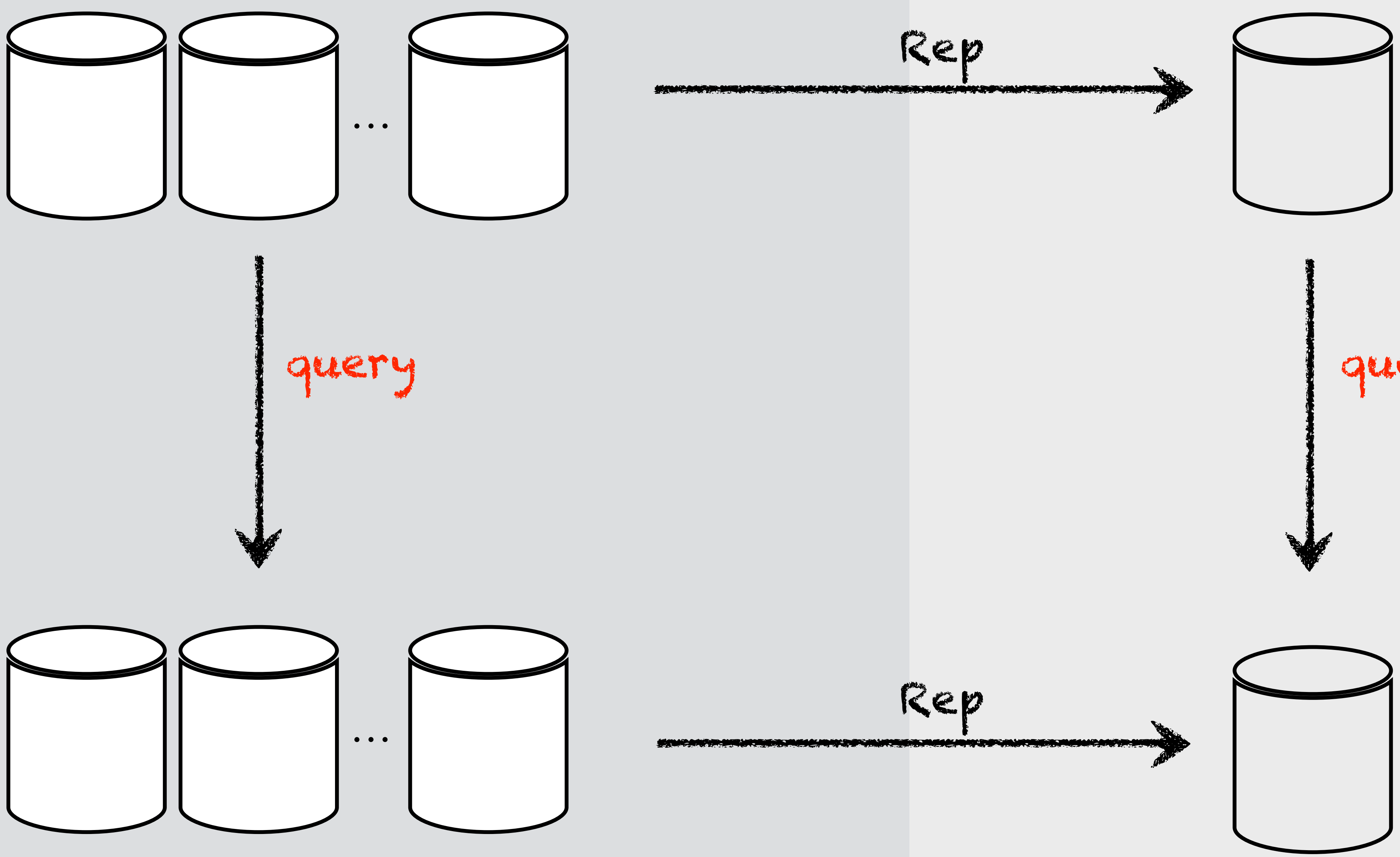
partial representation (c-tables)



difference (between regular- and c- tables) not visible to the query

loss-less modeling

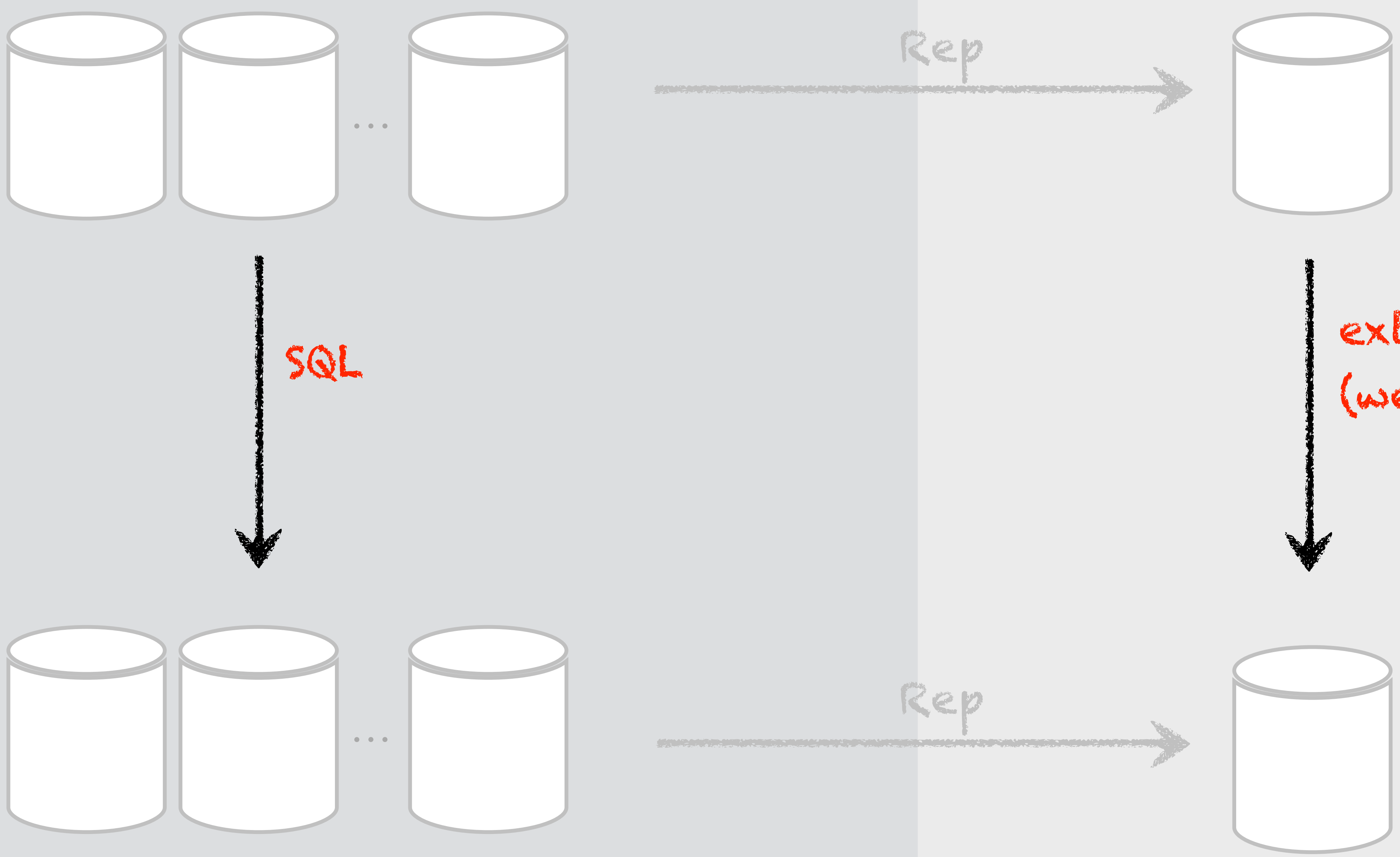
all definite instances (regular tables) partial representation (c-tables)



difference (between regular- and c- tables) not visible to the query

loss-less modeling with SQL?

all definite instances (regular tables) partial representation (c-tables)

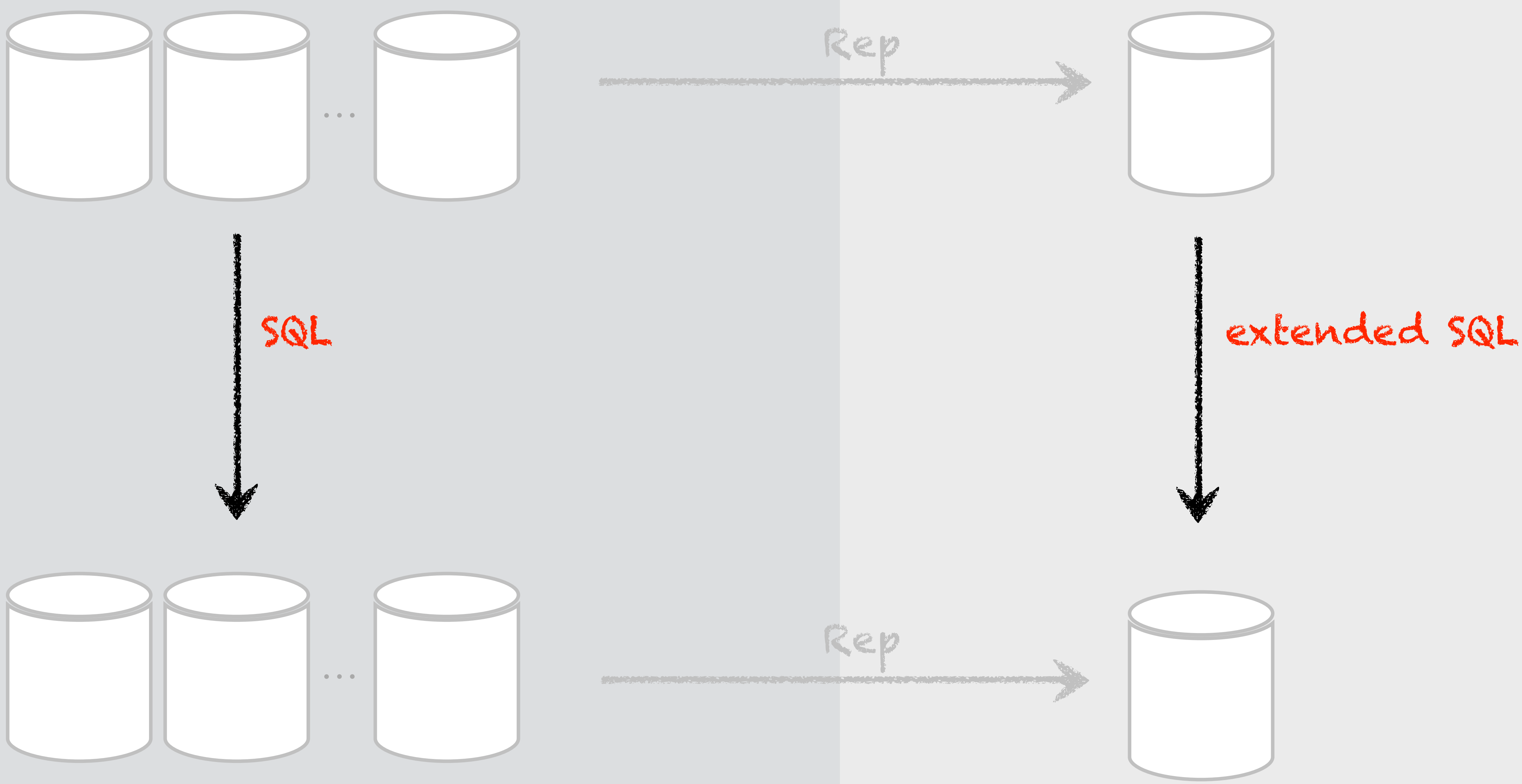


difference (between regular- and c- tables) not visible to the SQL

loss-less modeling with SQL?

all definite instances (regular tables)

partial representation (c-tables)



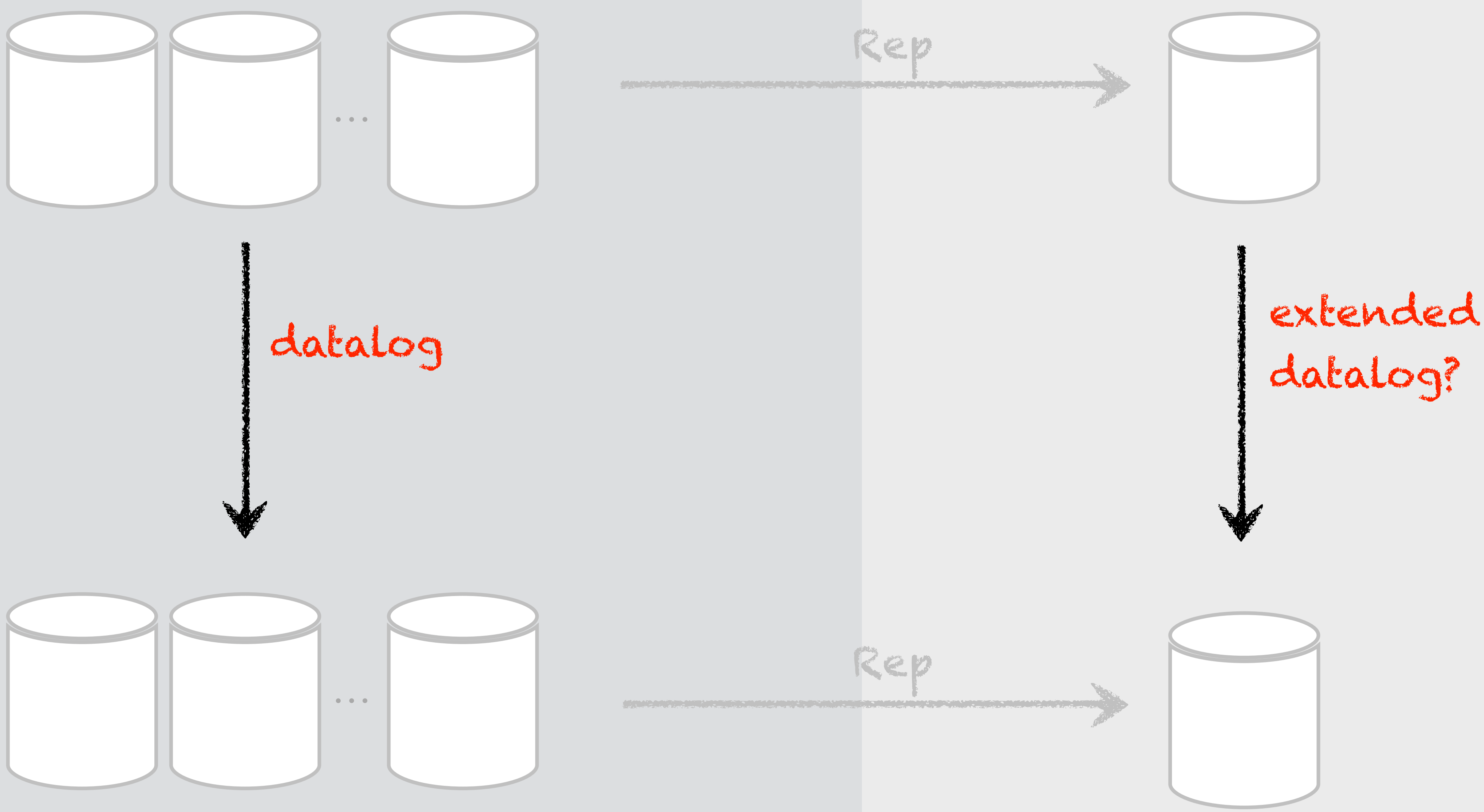
✓ ad hoc data retrieval

✗ static analysis

loss-less modeling with fauré-log

all definite instances (regular tables)

partial representation (c-tables)



✓ ad hoc data retrieval

✓ static analysis

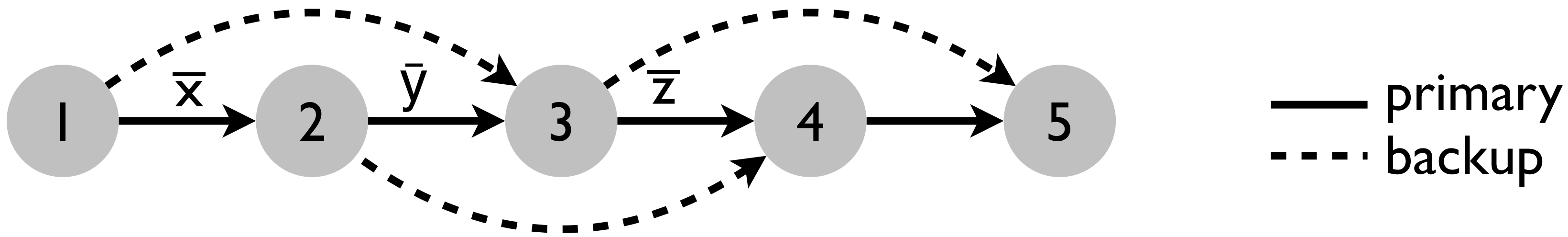
from datalog to fauré-log

	datalog	fauré-log
syntax (rules q)	$H(u) \text{ :- } B_1(u_1),\cdots,B_n(u_n).$	$H(u)[(\wedge^n_{i=1}\varphi_i)\wedge(\wedge^m_{i=1}C_i)] \text{ :- } B_1(u_1)[\varphi_1],\cdots,B_n(u_n)[\varphi_n], C_1,\cdots,C_m.$
semantics	$q(\mathbf{I}) = \{\upsilon(u) \mid \upsilon(u_i)\in\mathbf{I}\}$, $\mathbf{I}$ is a database over schema R	

notions and definitions

u_i (free tuples)	contains symbols in var(q) and dom(R)	
dom(R) (attribute domain over schema R)	constants	constants ∪ $\{\bar{x},\bar{y},\bar{z},\ldots\}$
υ (valuation)	υ: var(q)→dom(R) (i.e., $\{x,y,z,\ldots\}$ → constants ∪ $\{\bar{x},\bar{y},\bar{z},\ldots\}$)	
var(q) (variables)	$\{x,y,z,\ldots\}$	

Fauré-log queries



F	node	node	
	1	2	$\bar{x}=1$
	1	3	$\bar{x}=0$
	2	3	$\bar{y}=1$
	2	4	$\bar{y}=0$

…

recursive fauré-log

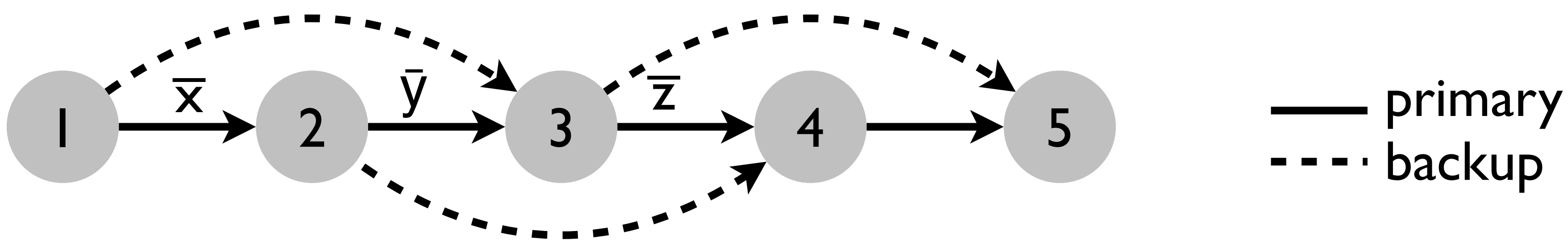
/* reachability query */

$$R(f,n_1,n_2)[\varphi] \text{ :- } F(f,n_1,n_2)[\varphi].$$

$$R(f,n_1,n_2)[\varphi_F \wedge \varphi_R] \text{ :- } F(f,n_1,n_3)[\varphi_F],\ R(f,n_3,n_2)[\varphi_R].$$

R	src	dest	
	1	2	$\bar{x}=1$
			…
	1	5	$\bar{x}=1 \wedge \bar{y}=1 \wedge \bar{z}=1$
	1	5	$\bar{x}=0 \wedge \bar{z}=1$
	1	5	$\bar{x}=0 \wedge \bar{z}=0$
	1	5	$\bar{x}=1 \wedge \bar{y}=0$
	2	3	$\bar{y}=1$
			…

Fauré-log queries

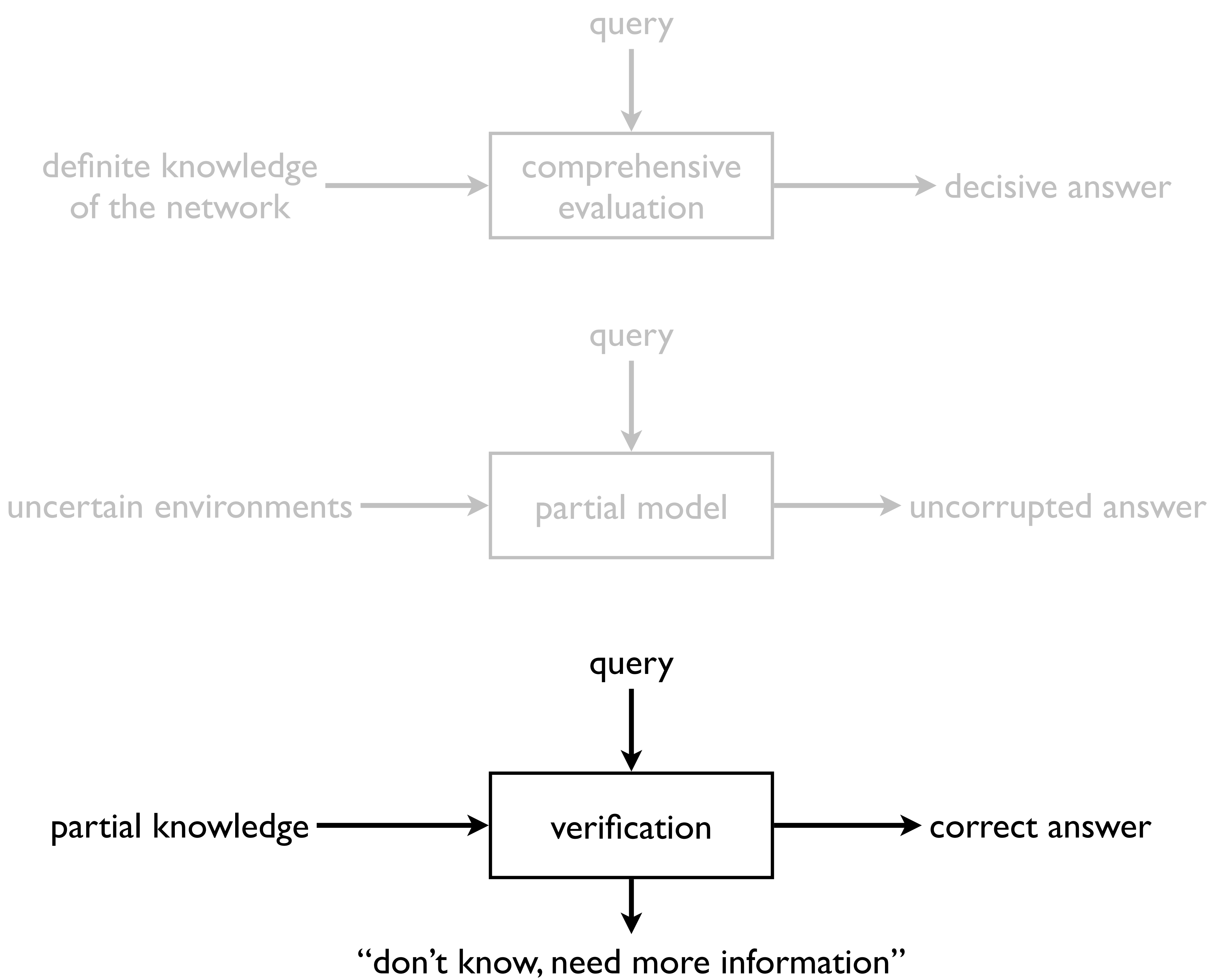


failure patterns over R

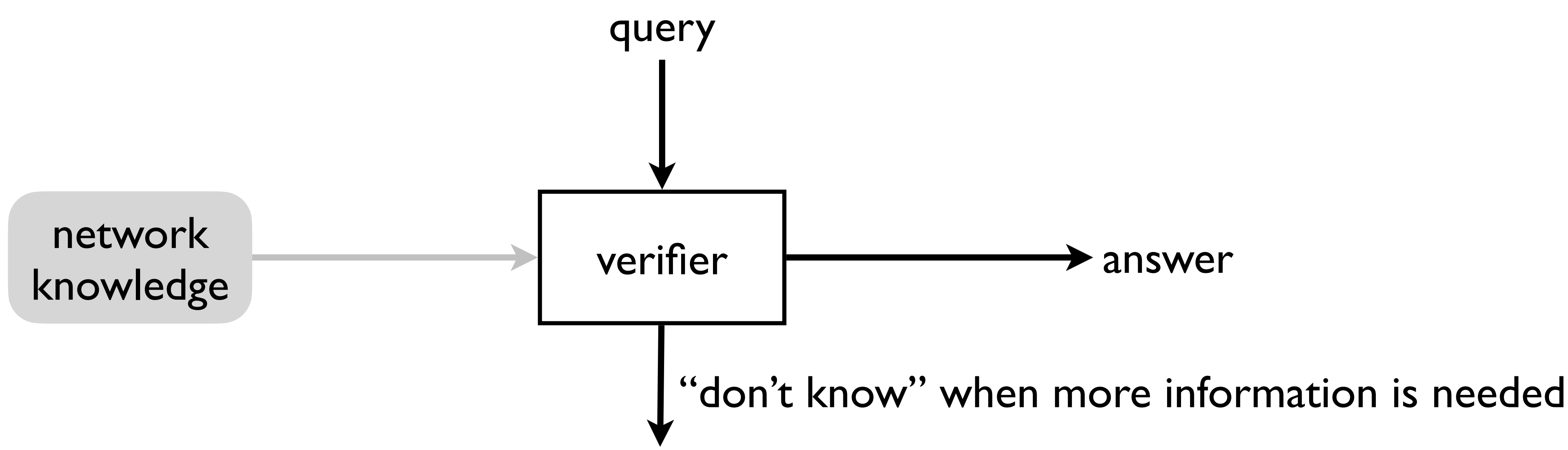
```
T1(f,n1,n2)[φ∧ x̄+ȳ+z̄=1] :- R(f,n1,n2)[φ],x̄+ȳ+z̄=1.
% reachability under 2-link failure

T2(f,2,5)[φ∧ ȳ=0] :- T1(f,2,5)[φ], ȳ=0. %
reachability between 2 and 5 under 2-link
failure, one of the failure must be (2,3)

T3(f,1,n2)[φ ∧ ȳ+z̄<2] :- R(f,1,n2)[φ], ȳ+z̄<2. %
reachability to 1 with at least 1-link failure
```

R	src	dest	
	1	2	$\bar{x}=1$
			…
	1	5	$\bar{x}=1 \wedge \bar{y}=1 \wedge \bar{z}=1$
	1	5	$\bar{x}=0 \wedge \bar{z}=1$
	1	5	$\bar{x}=0 \wedge \bar{z}=0$
	1	5	$\bar{x}=1 \wedge \bar{y}=0$
	2	3	$\bar{y}=1$
			…

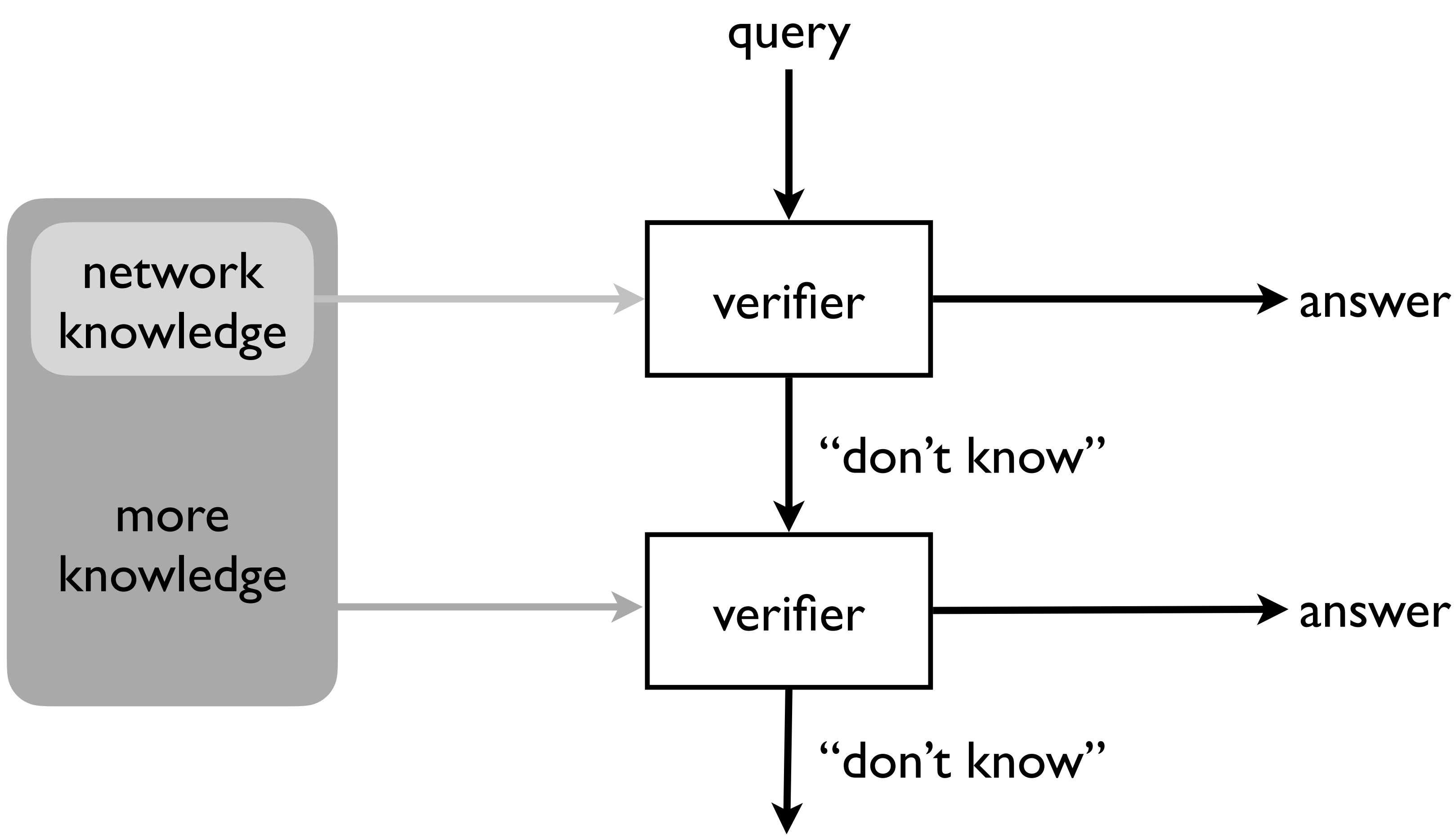
relative-complete verification



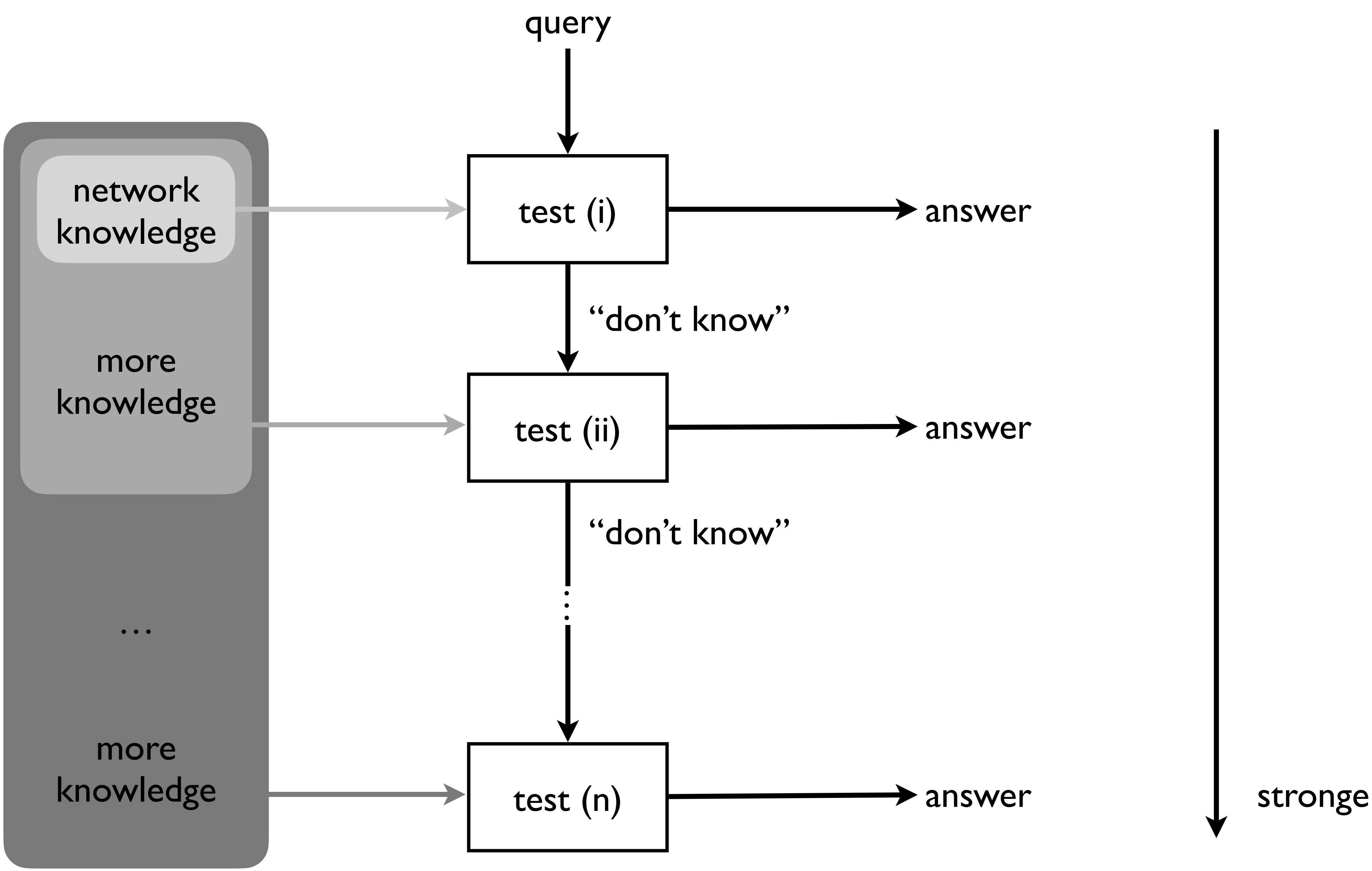
relative-complete verification



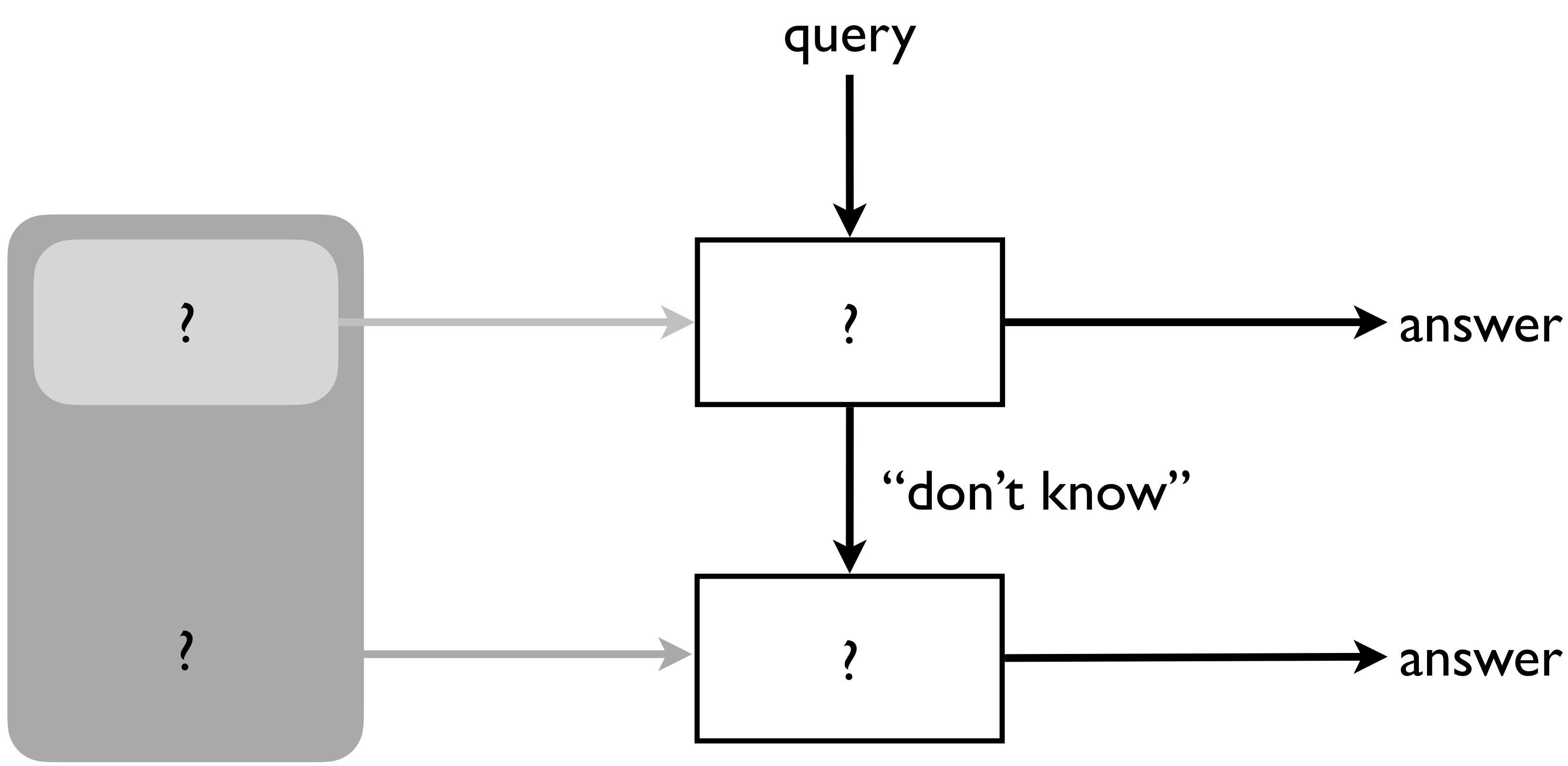
relative-complete verification

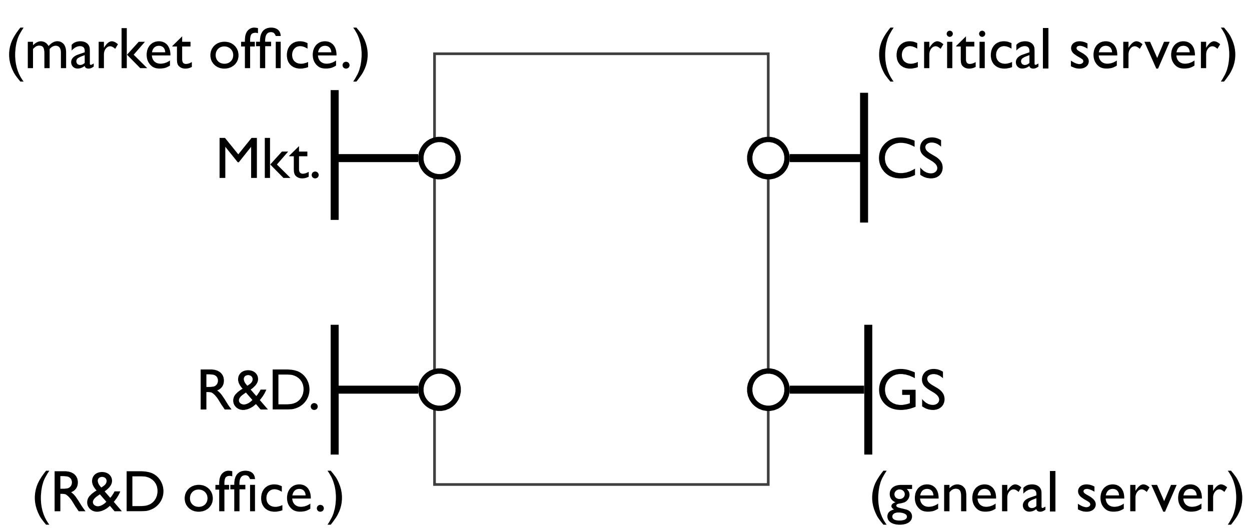


relative-complete verification



example relative-complete verification





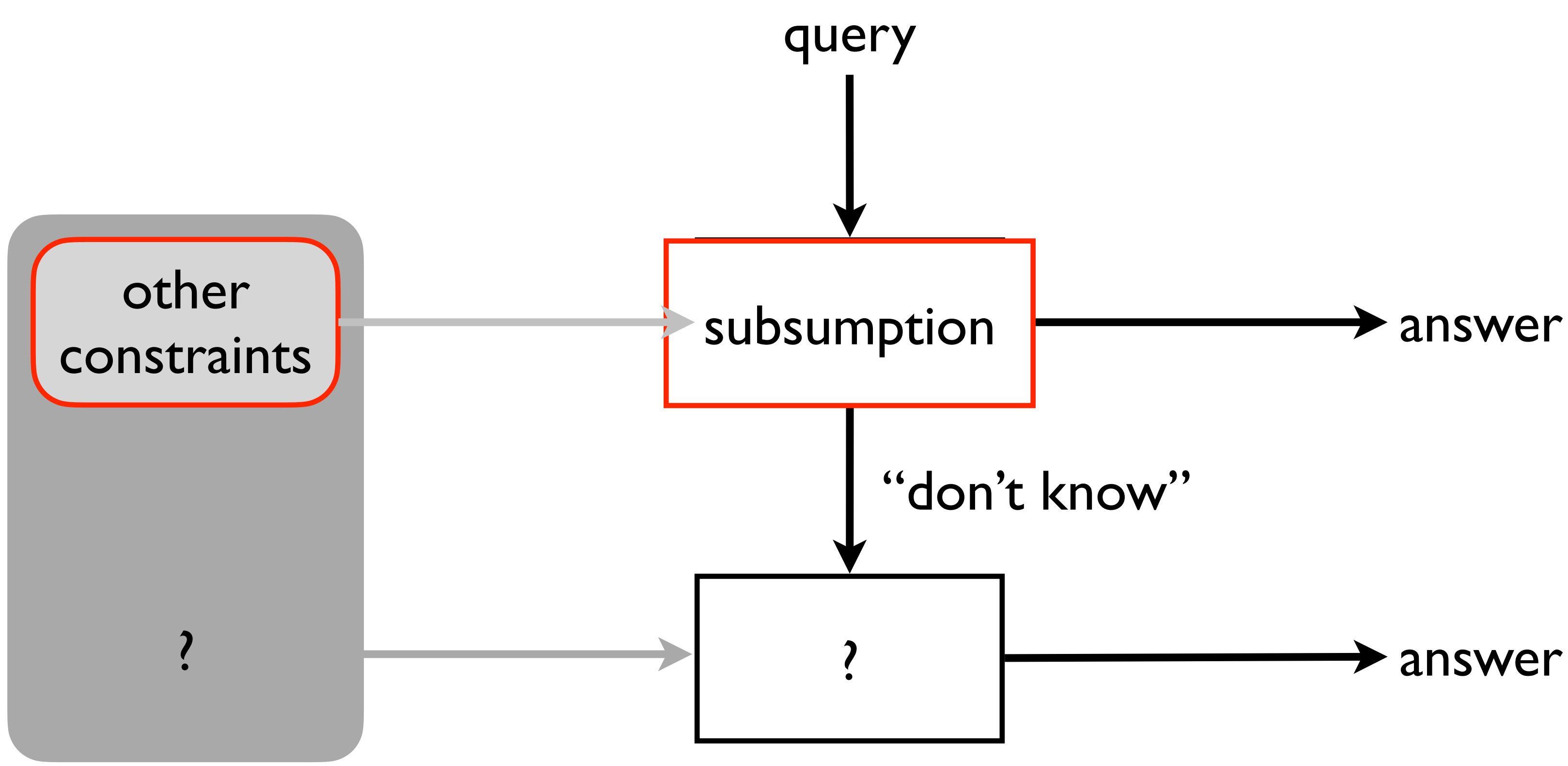
verification task

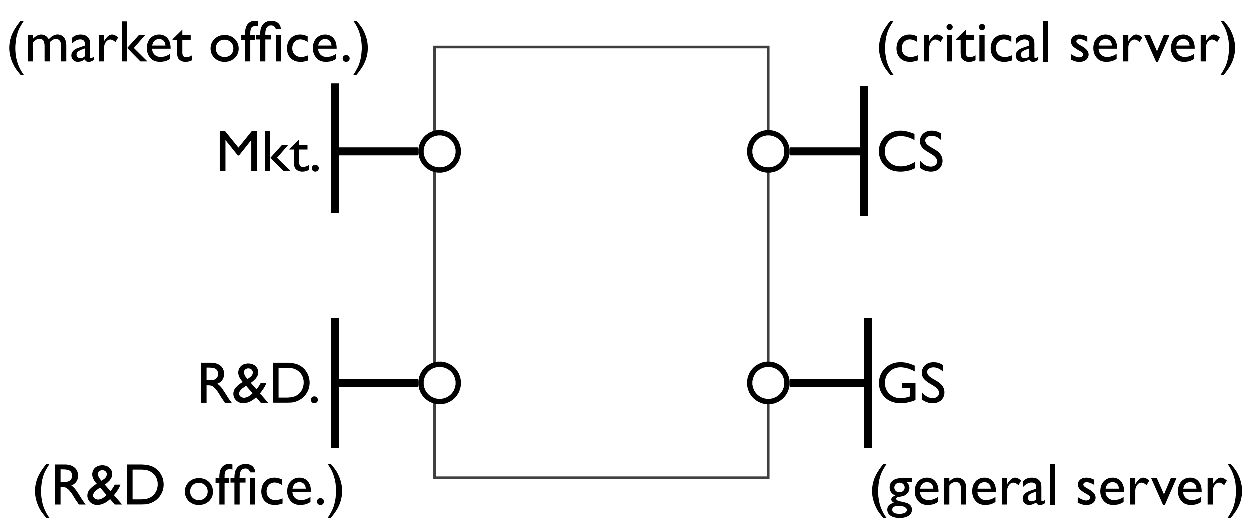
- invariants (T_1,T_2) continue to hold after updates

other teams

- security team maintains C_S
- TE team maintains C_{lb}

example relative-complete verification





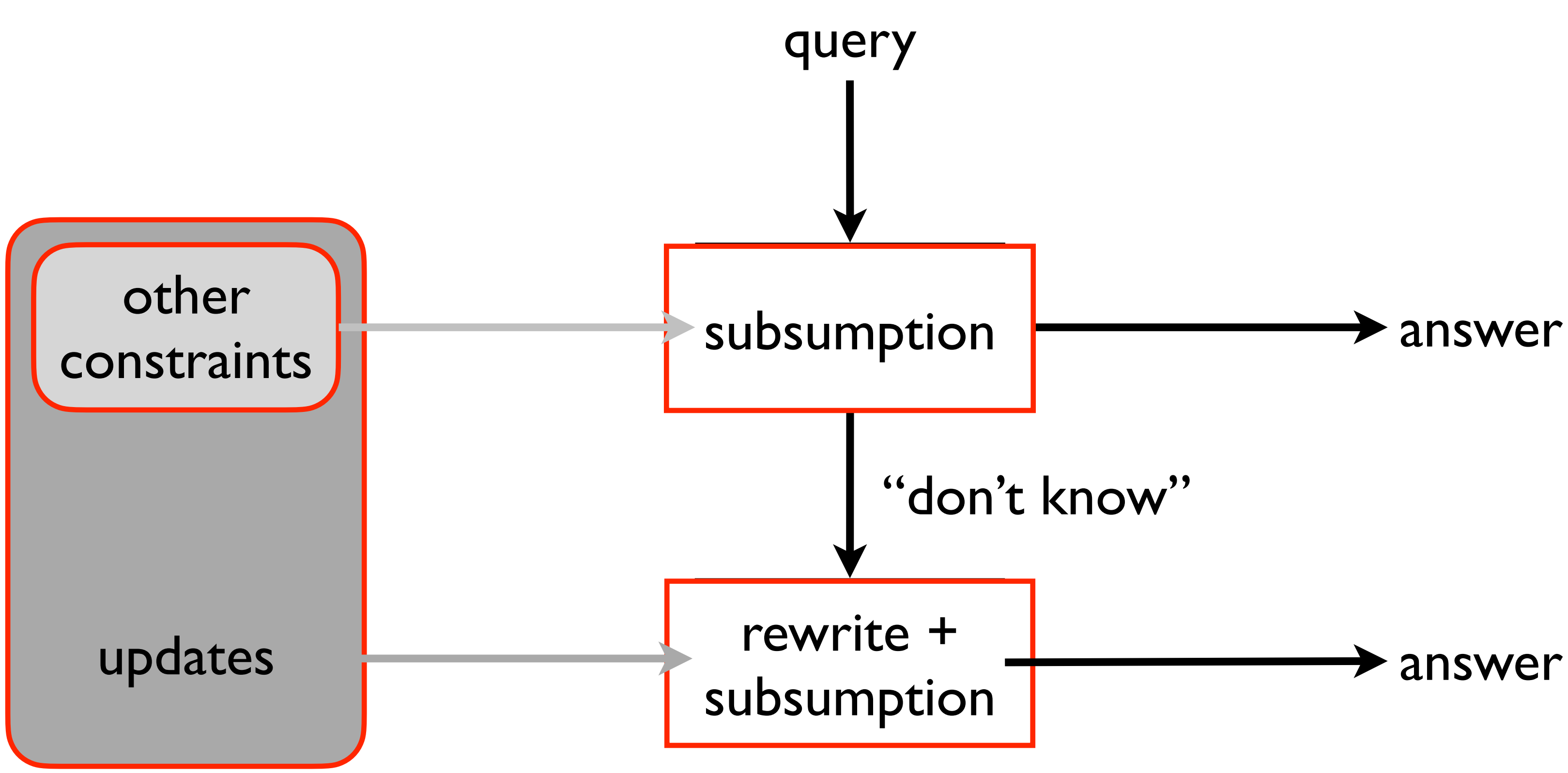
verification task

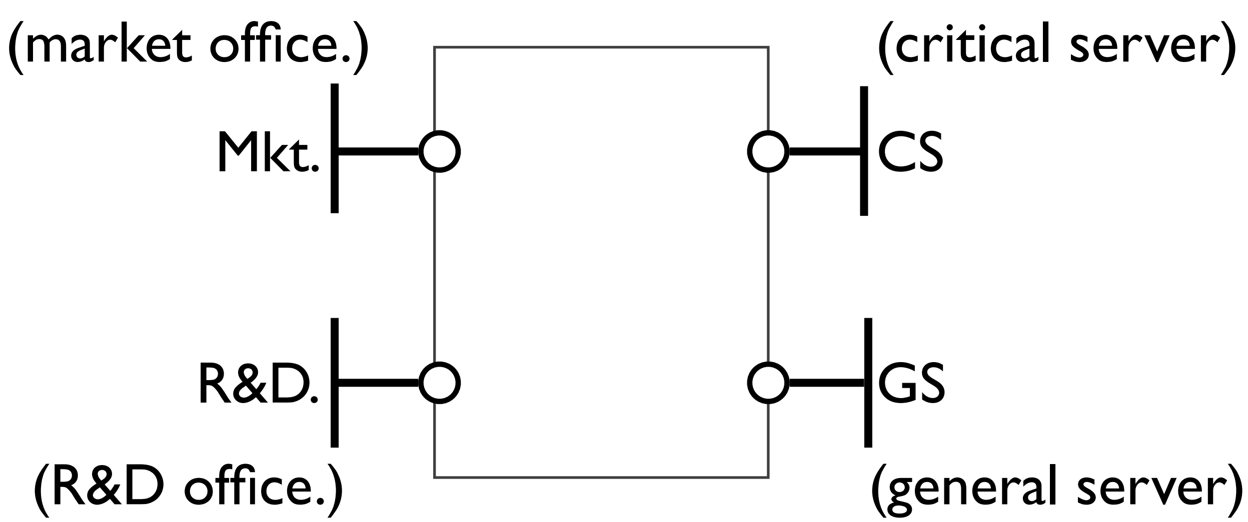
- invariants (T_1,T_2) continue to hold after updates

other teams

- security team maintains C_S
- TE team maintains C_{lb}

example relative-complete verification





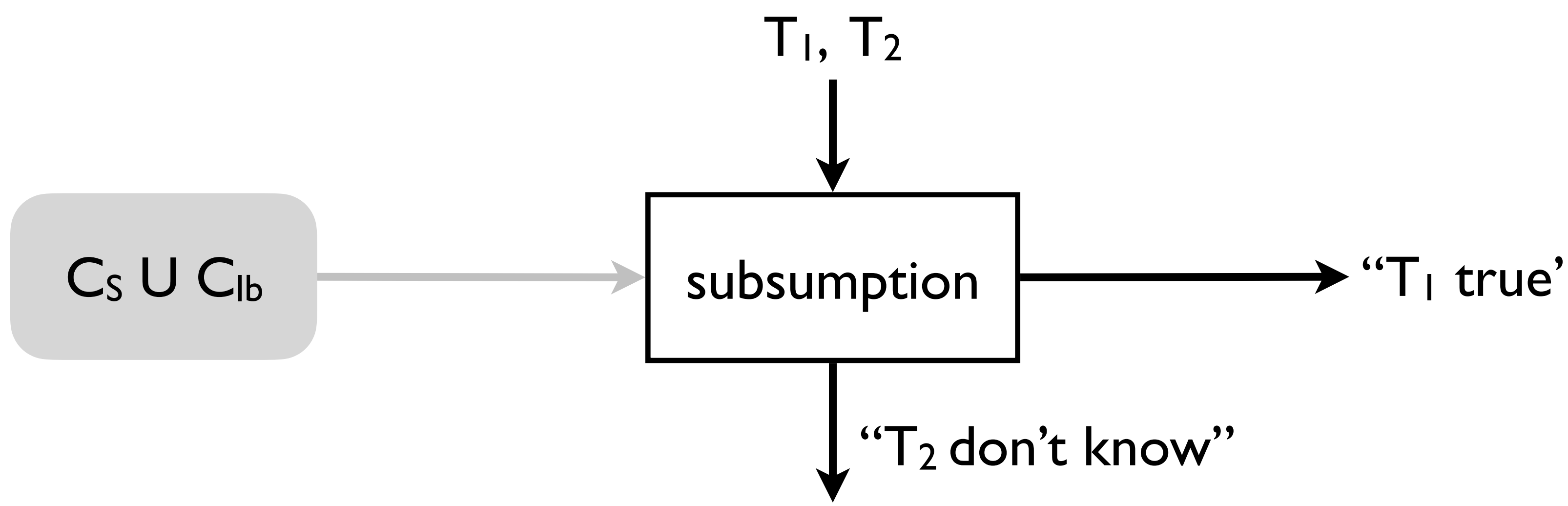
verification task

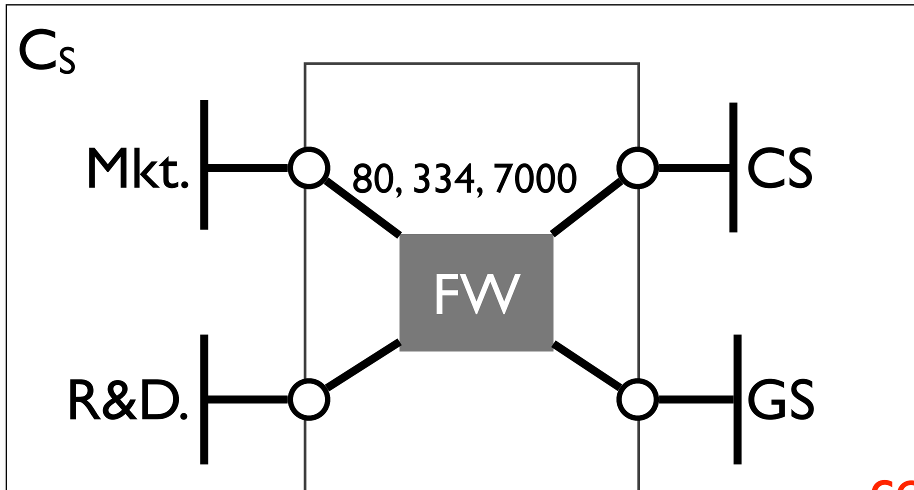
- invariants (T_1, T_2) continue to hold after updates

other teams

- security team maintains C_S
- TE team maintains C_{lb}

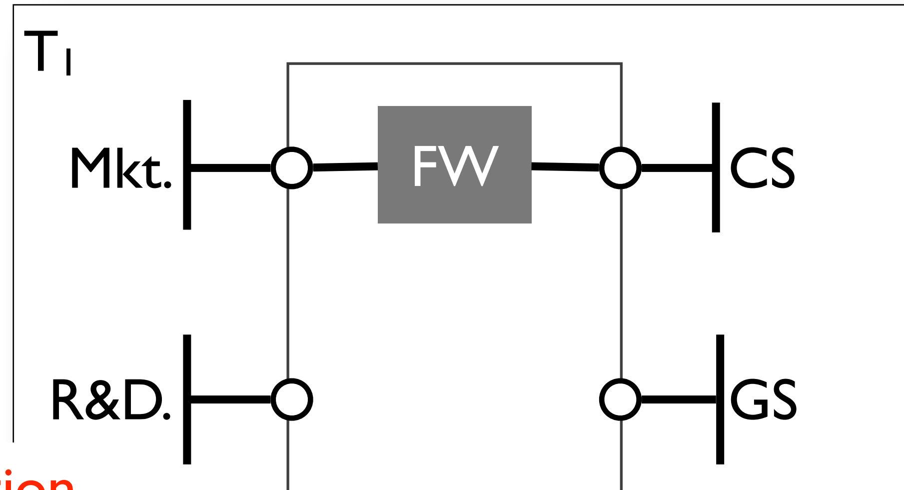
category (i) test: using only constraints





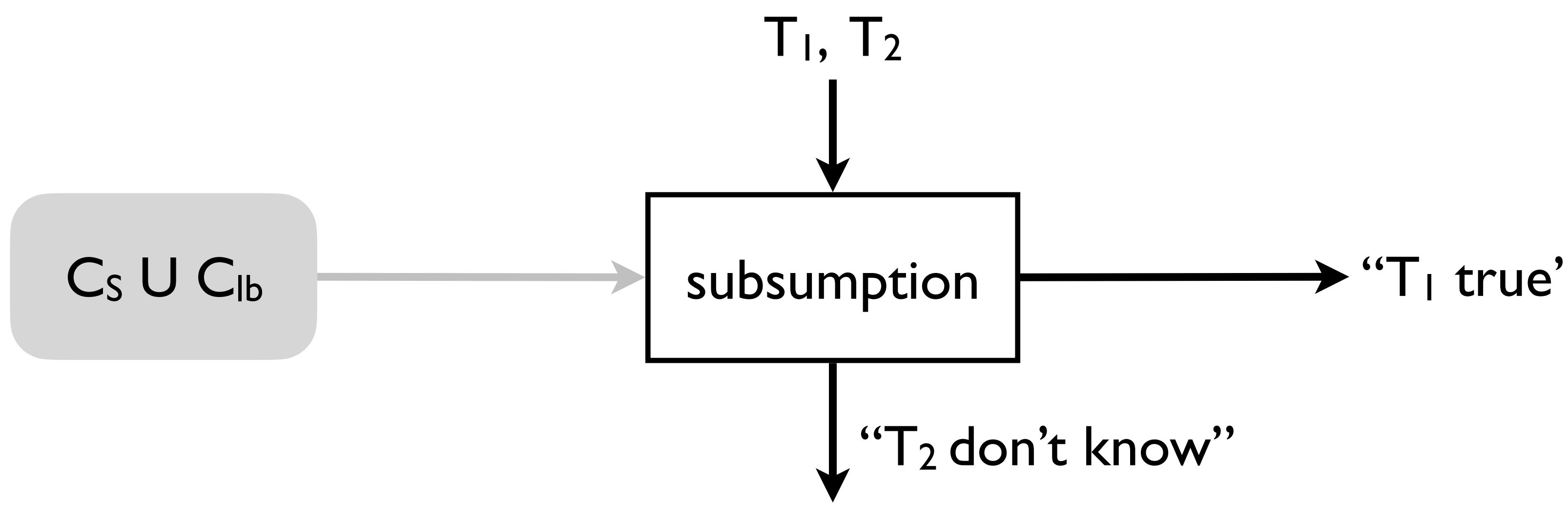
packets to all the servers, must use one of the three ports 80, 334 and 7000, and must pass through a firewall

constraint subsumption

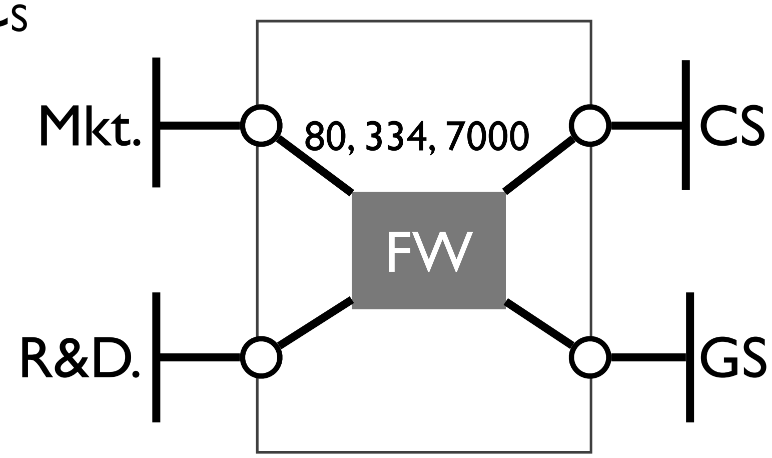


Mkt traffic to the critical server CS to go through a firewall

category (i) test: using only constraints



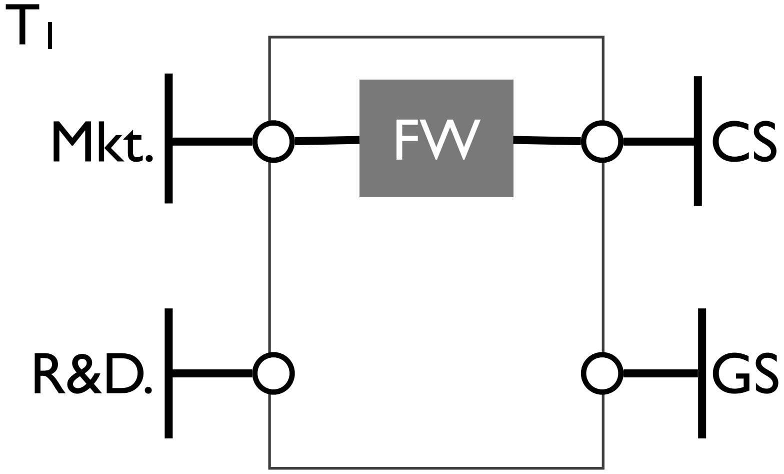
Cs




```
panic :- Vs(x,y,p)

Vs(x̄,ȳ,p̄) :- R(x̄,ȳ,p̄),
         ¬Fw(x̄,ȳ).
Vs(x̄,ȳ,p̄) :- R(x̄,ȳ,p̄),
         p̄≠80,p̄≠344,
         p̄≠7000.
```

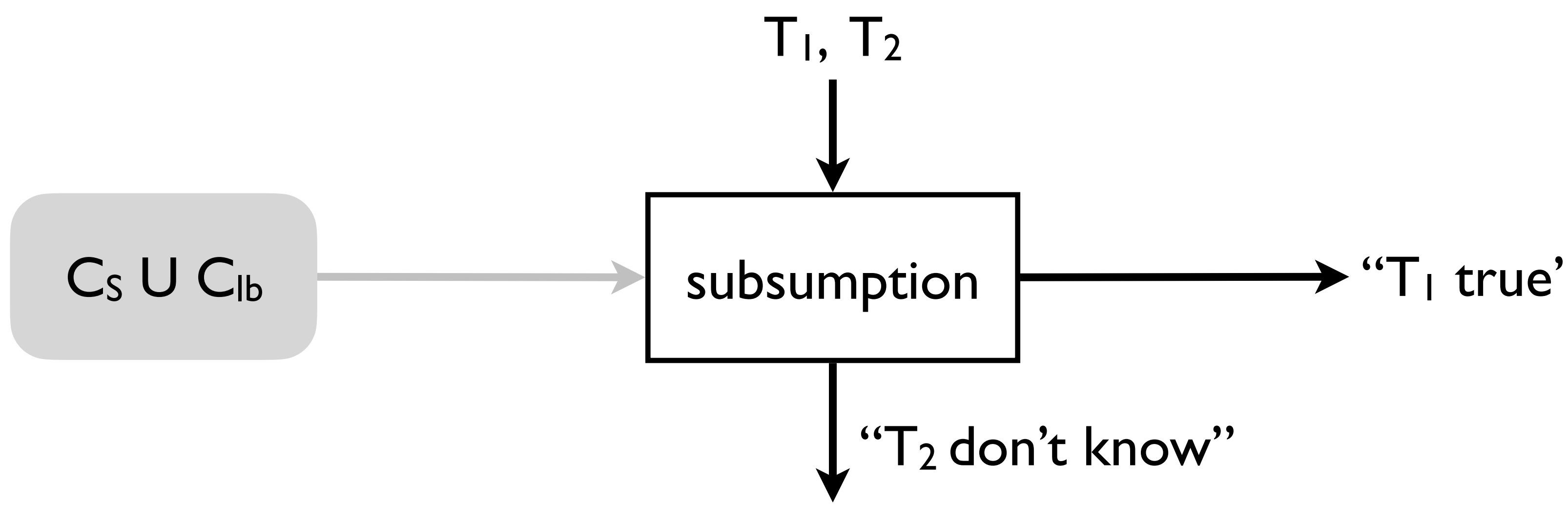
constraint as
0-ary fauré-log
query (panic)

T1

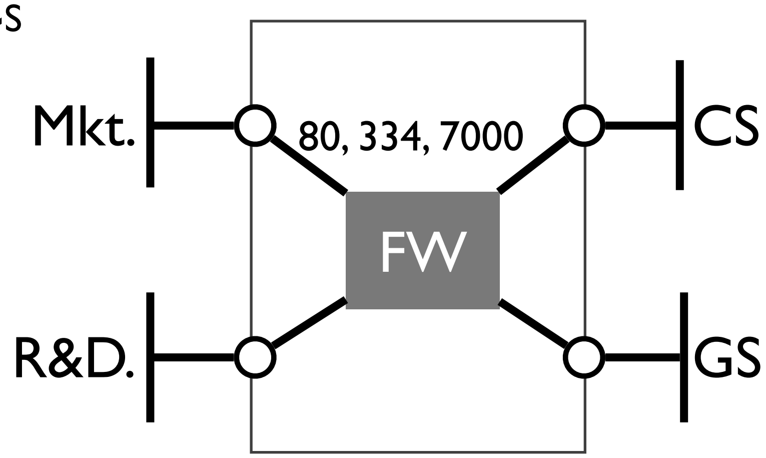



```
panic :- R(Mkt,CS,p̄),
         ¬Fw(Mkt,CS)
```

category (i) test: using only constraints



C_S

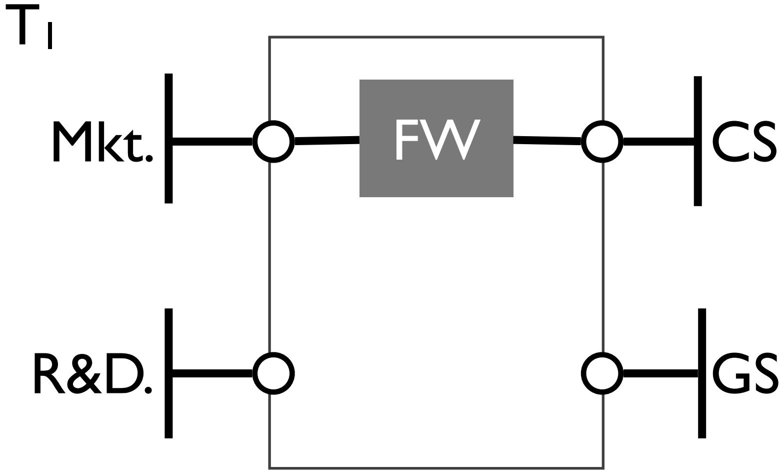


$\texttt{panic :- } V_s(x,y,p)$

$V_s(\bar{x},\bar{y},\bar{p}) \texttt{ :- } R(\bar{x},\bar{y},\bar{p}),$
$\neg Fw(\bar{x},\bar{y}).$

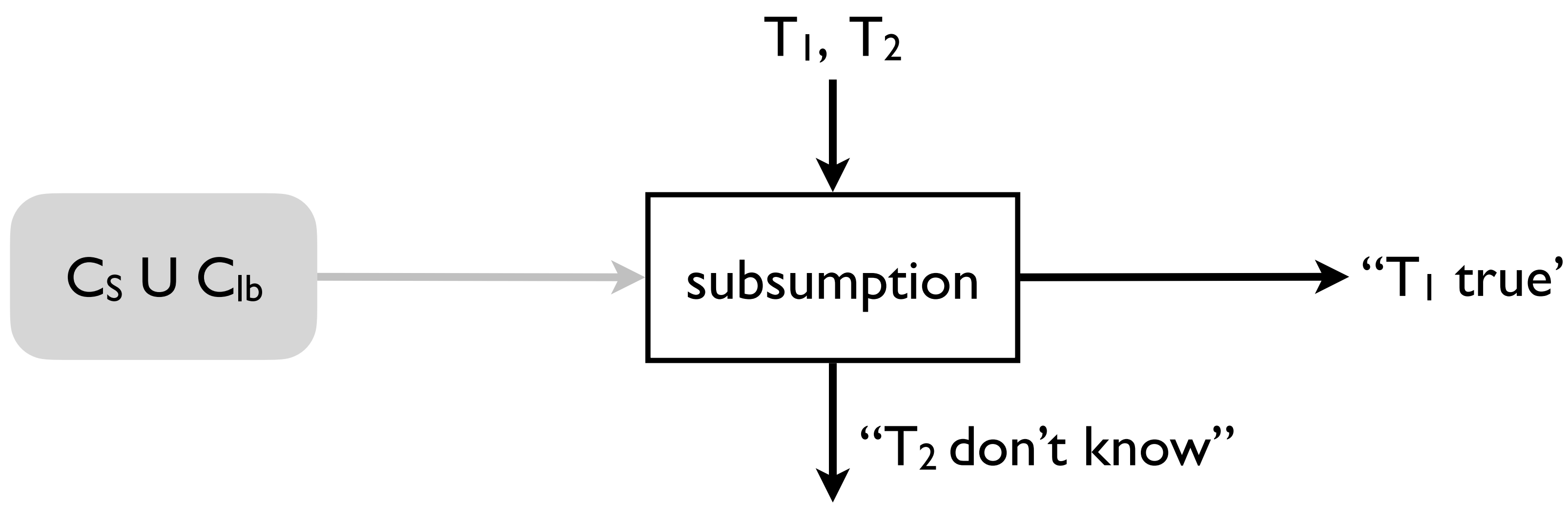
$V_s(\bar{x},\bar{y},\bar{p}) \texttt{ :- } R(\bar{x},\bar{y},\bar{p}),$
$\bar{p} \neq 80, \bar{p} \neq 344,$
$\bar{p} \neq 7000.$

T_1

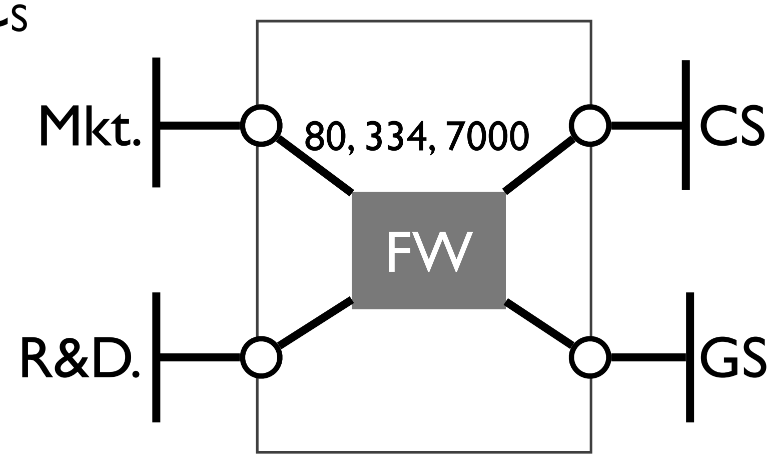


$\texttt{panic :- } R(Mkt,CS,\bar{p}),$
$\neg Fw(Mkt,CS)$

category (i) test: using only constraints



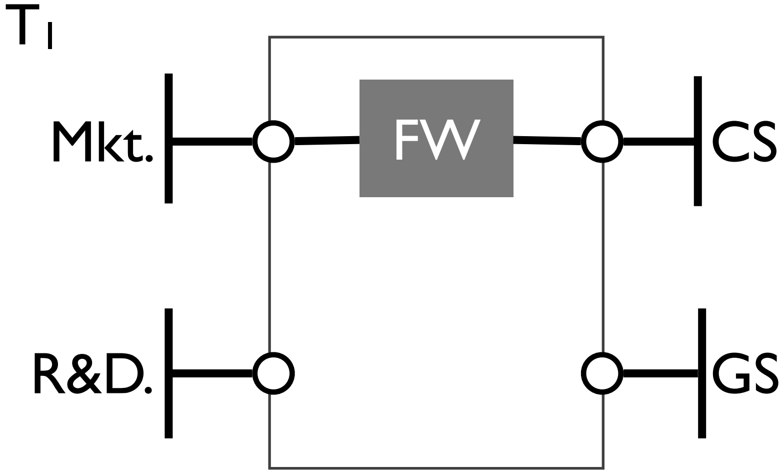
Cs




```
panic :- Vs(x,y,p)
Vs(x̄,ȳ,p̄) :- R(x̄,ȳ,p̄),
        ¬Fw(x̄,ȳ).
Vs(x̄,ȳ,p̄) :- R(x̄,ȳ,p̄),
        p̄≠80,p̄≠344,
        p̄≠7000.
```

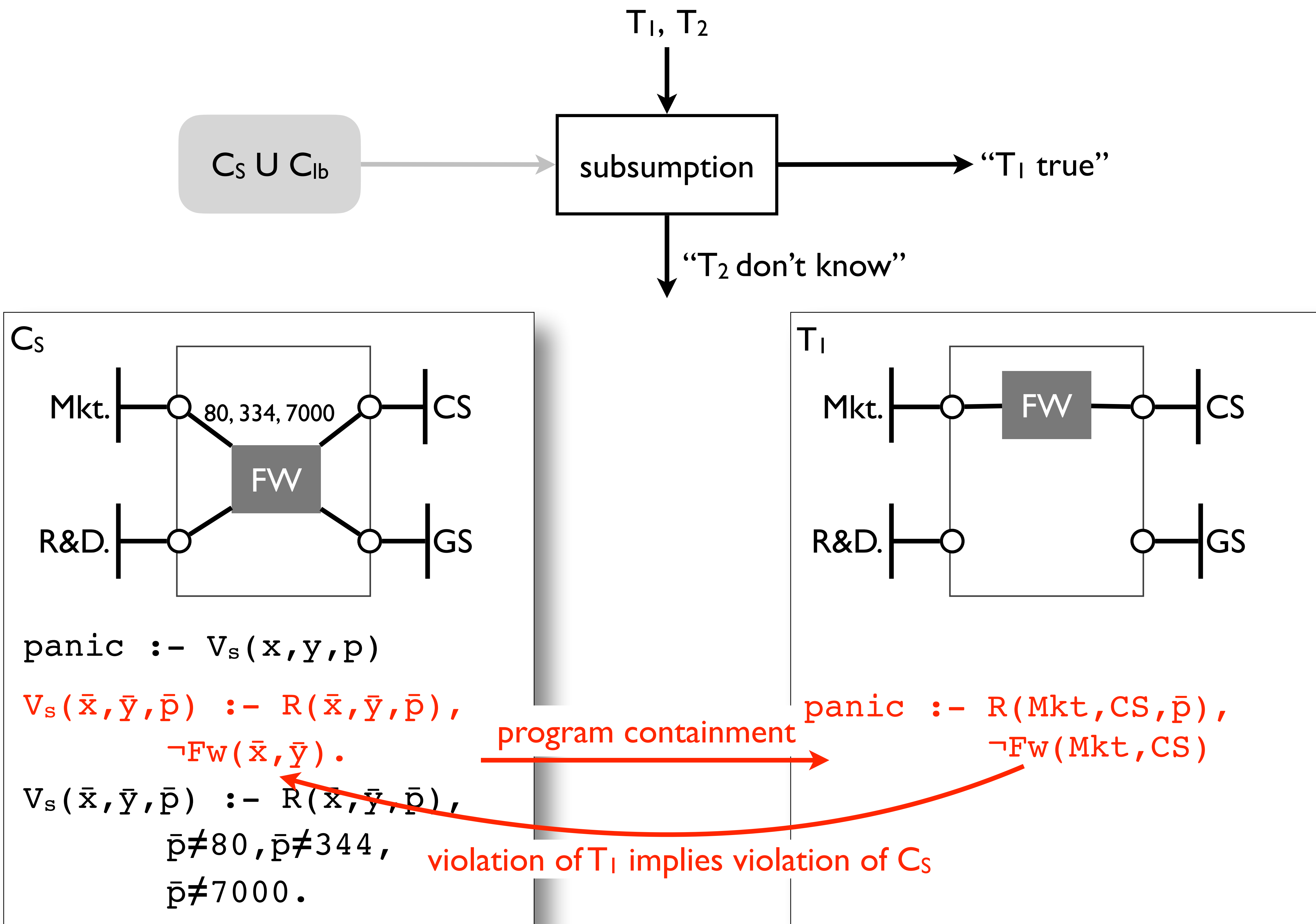
program containment

T1

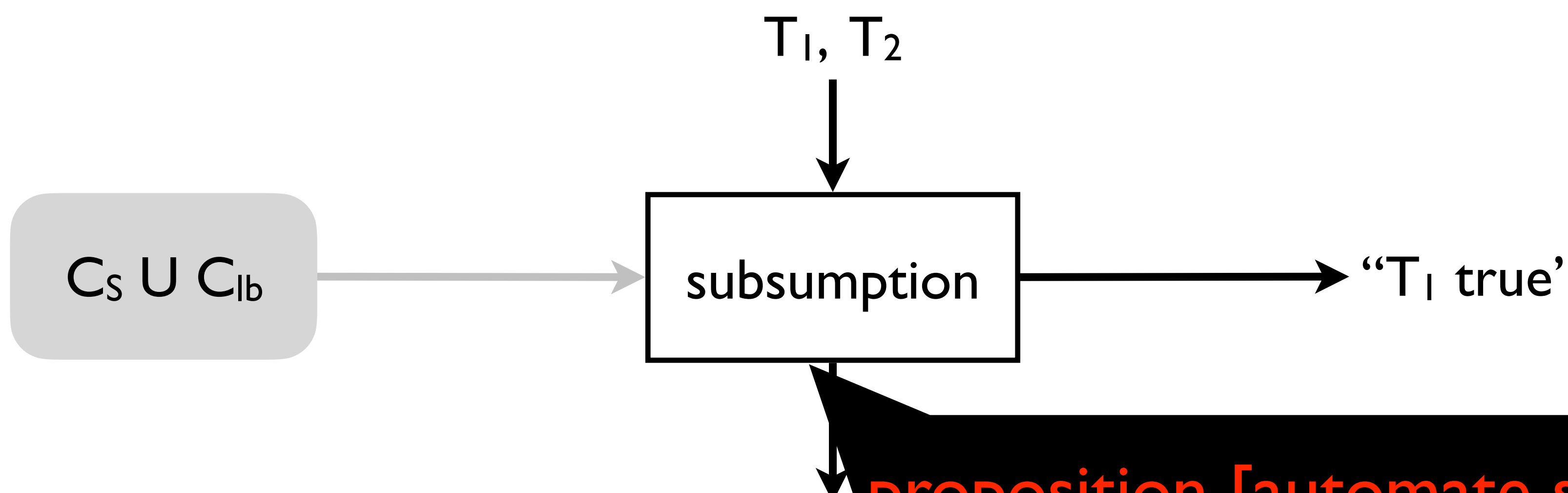



```
panic :- R(Mkt,CS,p̄),
         ¬Fw(Mkt,CS)
```

category (i) test: using only constraints



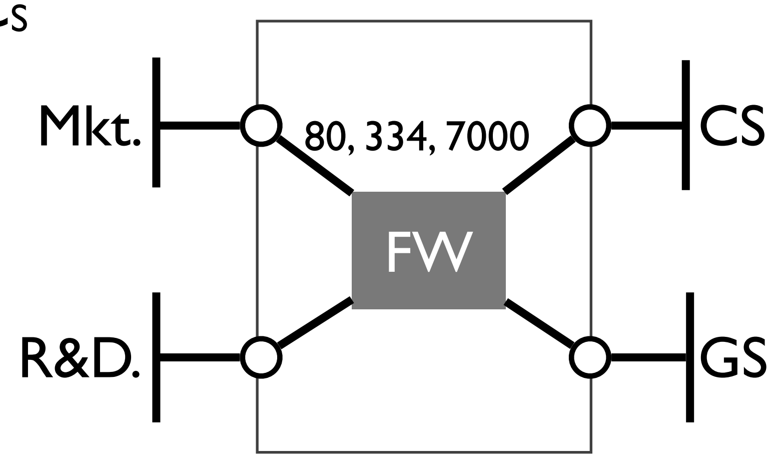
category (i) test: using only constraints



proposition [automate subsumption]

$\forall p_1, p_2$. p_1 and p_2 are two constraint programs, then $p_1 \supseteq p_2$ (p_1 contains/implies p_2) if applying p_1 (i.e., fauré-log valuation) to the "instance" of p_2 yeilds `panic`

C_S



$$\texttt{panic :- } V_s(x,y,p)$$

$$V_s(\bar{x},\bar{y},\bar{p}) \texttt{ :- } R(\bar{x},\bar{y},\bar{p}),\ \neg Fw(\bar{x},\bar{y}).$$

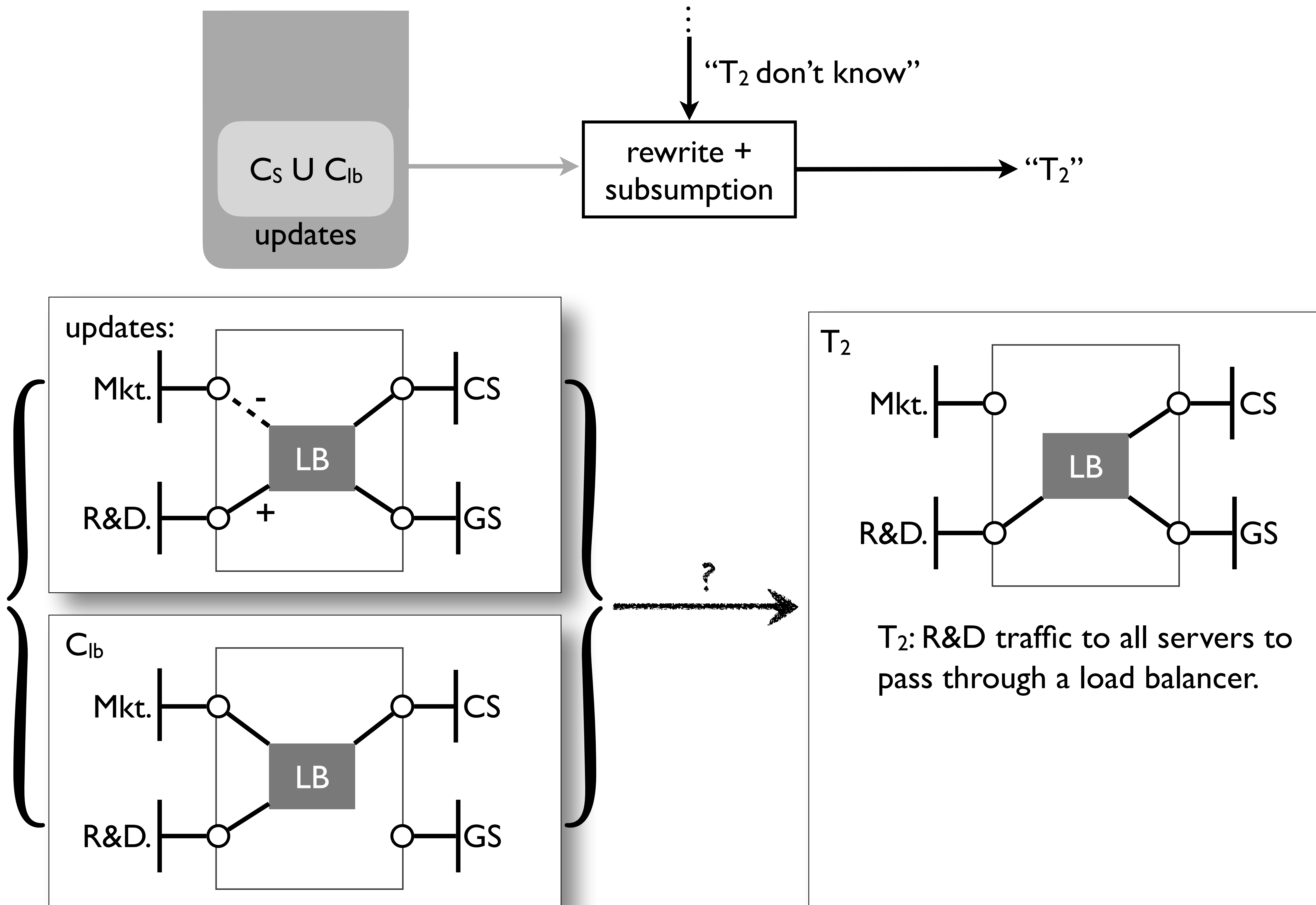
$$V_s(\bar{x},\bar{y},\bar{p}) \texttt{ :- } R(\bar{x},\bar{y},\bar{p}),\ \bar{p}\neq 80, \bar{p}\neq 344,\ \bar{p}\neq 7000.$$

program containment

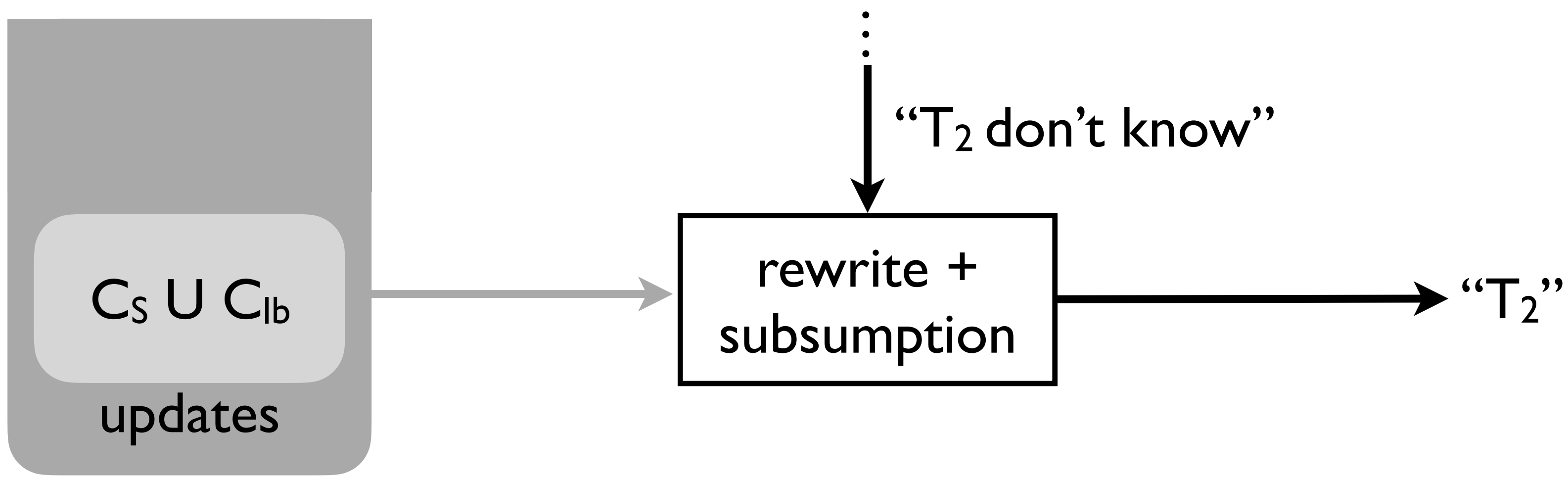
$$\texttt{panic :- } R(Mkt,CS,\bar{p}),\ \neg Fw(Mkt,CS)$$

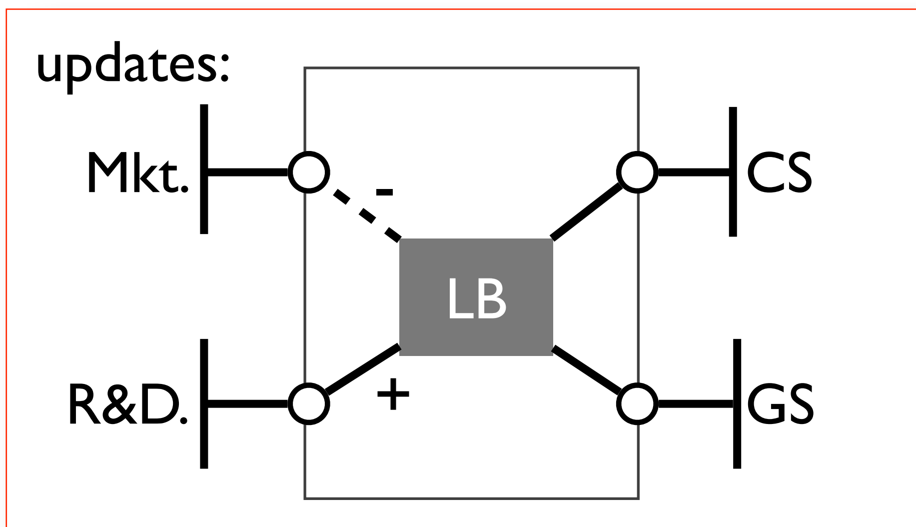
violation of T_1 implies violation of C_S

category (ii) test: using constraints & updates

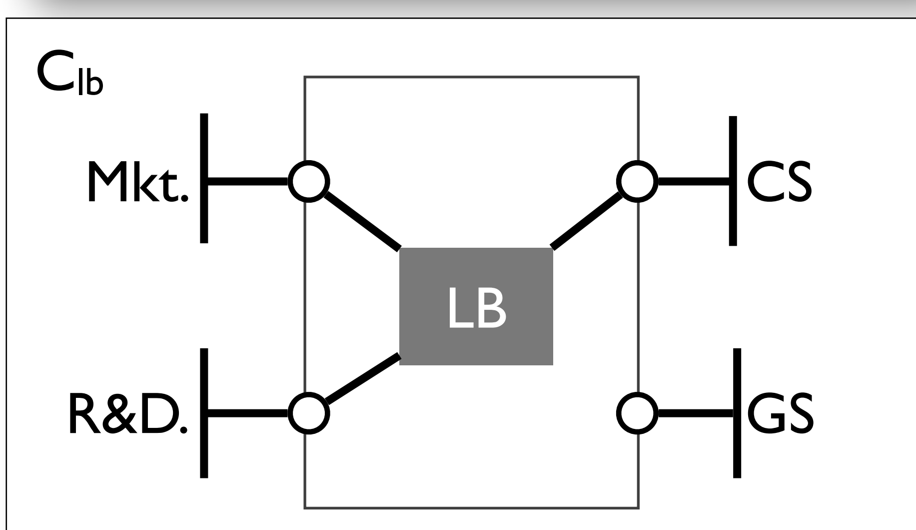


category (ii) test: using constraints & updates

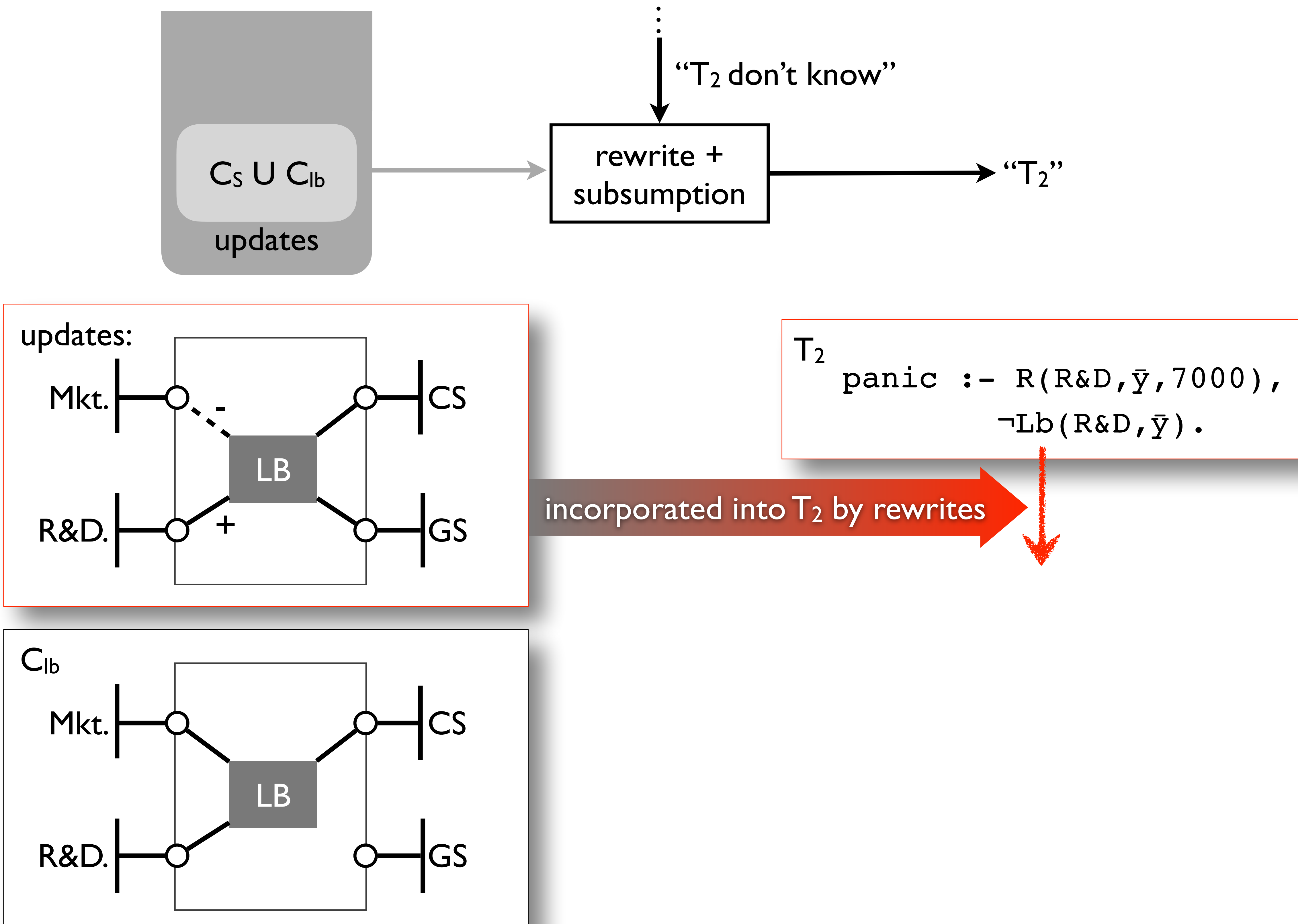




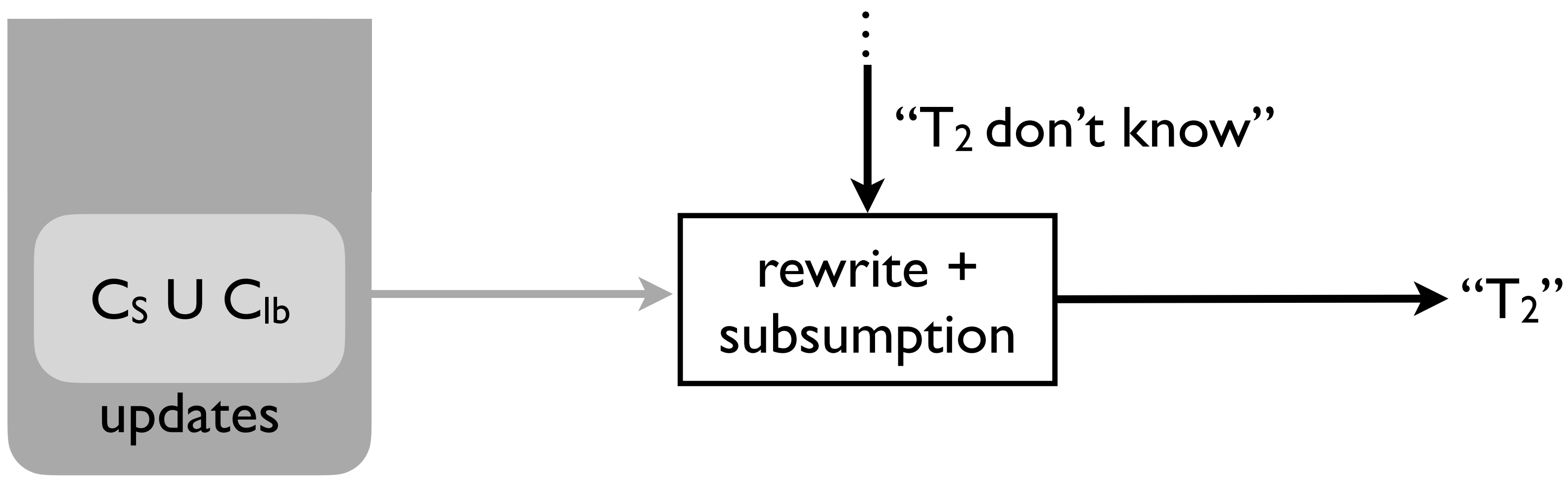
T2
```
panic :- R(R&D,ȳ,7000),
         ¬Lb(R&D,ȳ).
```

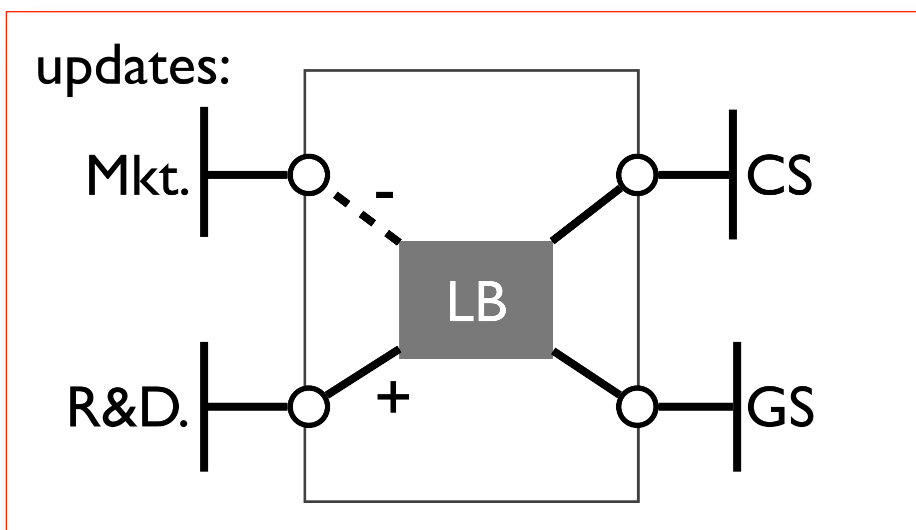


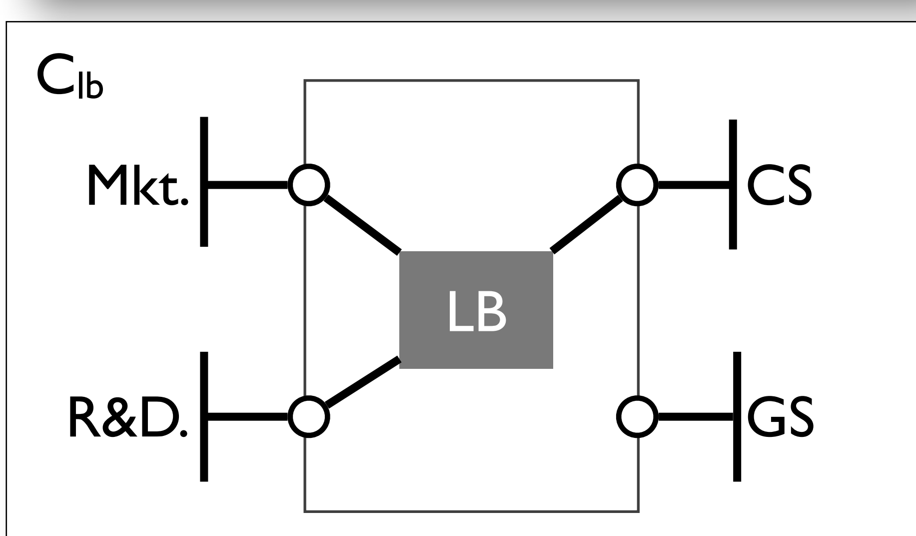
category (ii) test: using constraints & updates



category (ii) test: using constraints & updates







incorporated into T_2 by rewrites

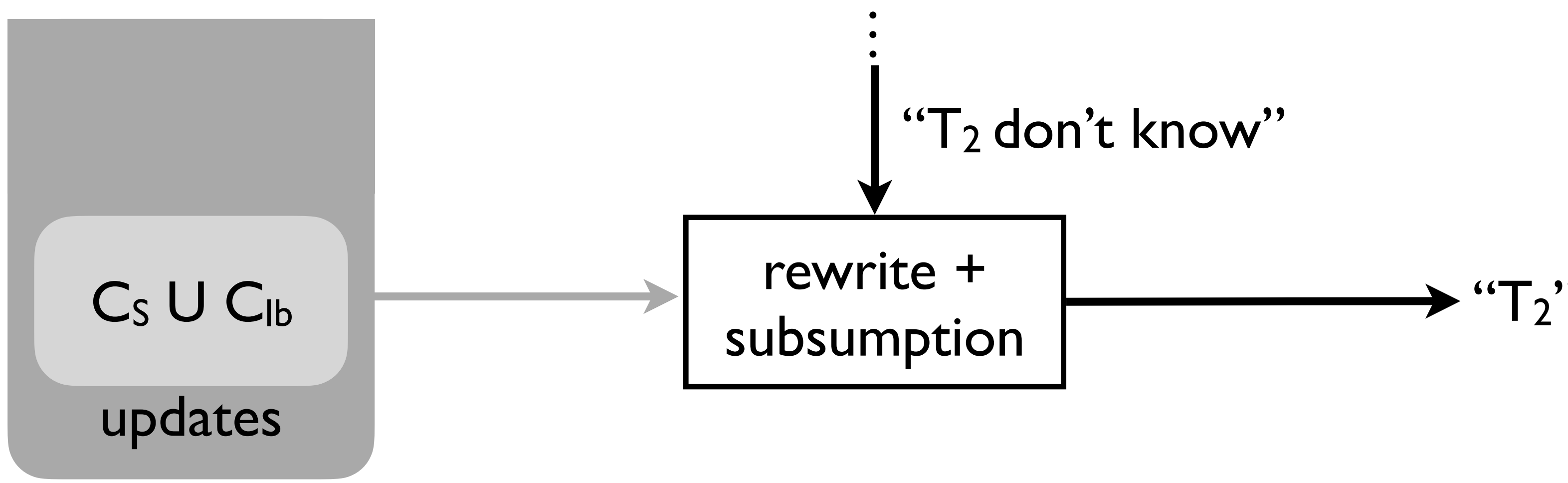
T_2

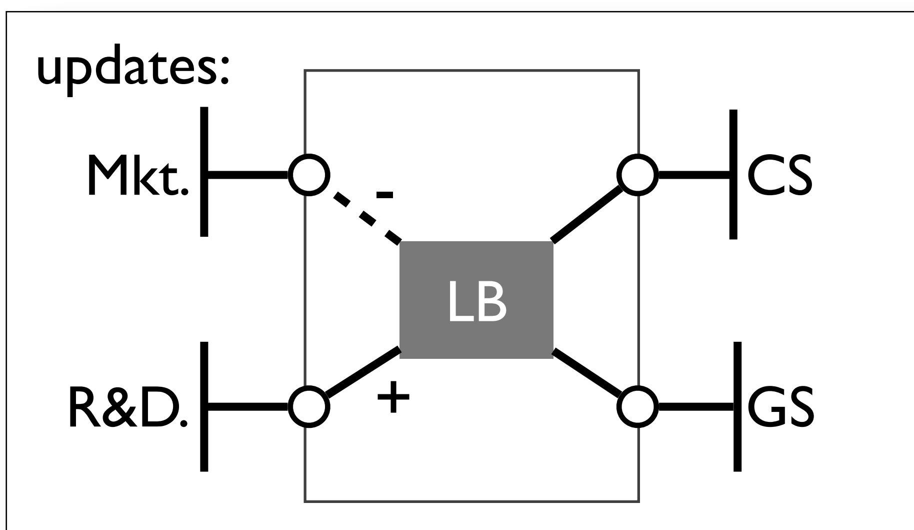
```
panic :- R(R&D,ȳ,7000),
         ¬Lb(R&D,ȳ).
```

T_2'

```
/* add (R&D,GS) to LB */
Lb(R&D,GS).
Lb1(x̄,ȳ) :- Lb(x̄,ȳ)

/* delete (Mkt,CS) LB */
Lb2(x̄,ȳ):- Lb1(x̄,ȳ)[x̄≠Mkt]
Lb2(x̄,ȳ) :- Lb1(x̄,ȳ)[ȳ≠CS]

/*panic after updates */
panic :- R(R&D,ȳ,7000),
         ¬Lb2(R&D,ȳ)
```

category (ii) test: using constraints & updates





C_{lb}

```
panic :- Vt(x,y,p)

Vt(x̄,CS,p̄) :-
R(x̄,CS,p̄),x̄≠Mkt,x̄≠R&D.

Vt(x̄,CS,p̄) :-
R(x̄,CS,p̄),¬Lb(x̄,CS)

Vt(x̄,CS,p̄) :-
R(x̄,CS,p̄),p̄≠7000
```

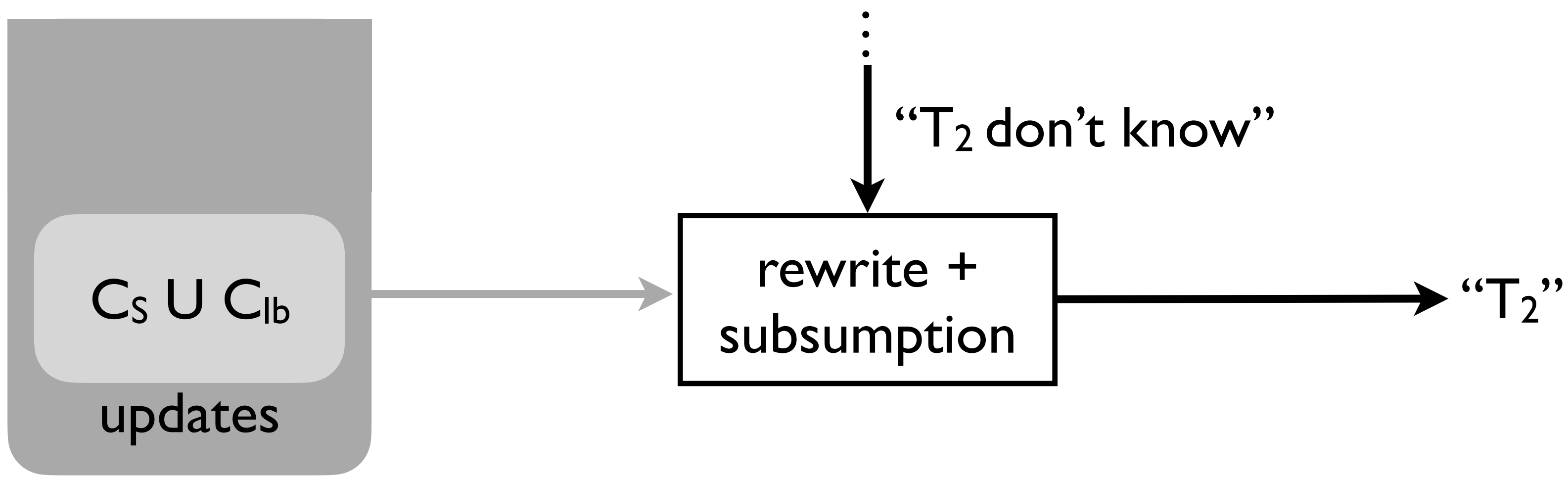
T_2

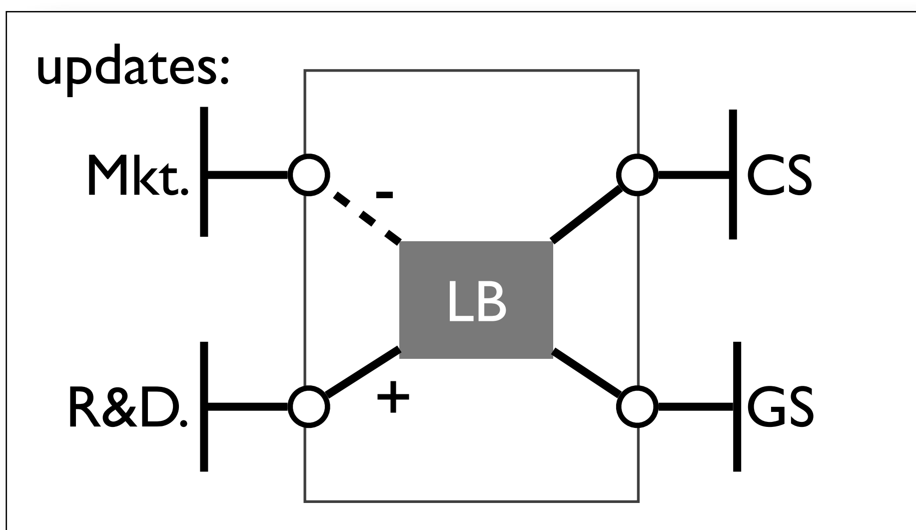
```
panic :- R(R&D,ȳ,7000),
         ¬Lb(R&D,ȳ).
```

T_2'

```
/* add (R&D,GS) to LB */
Lb(R&D,GS).
Lb1(x̄,ȳ) :- Lb(x̄,ȳ)

/* delete (Mkt,CS) LB */
Lb2(x̄,ȳ):- Lb1(x̄,ȳ)[x̄≠Mkt]
Lb2(x̄,ȳ) :- Lb1(x̄,ȳ)[ȳ≠CS]

/*panic after updates */
panic :- R(R&D,ȳ,7000),
         ¬Lb2(R&D,ȳ)
```

category (ii) test: using constraints & updates





T_2

```
panic :- R(R&D,ȳ,7000),
         ¬Lb(R&D,ȳ).
```

C_{lb}

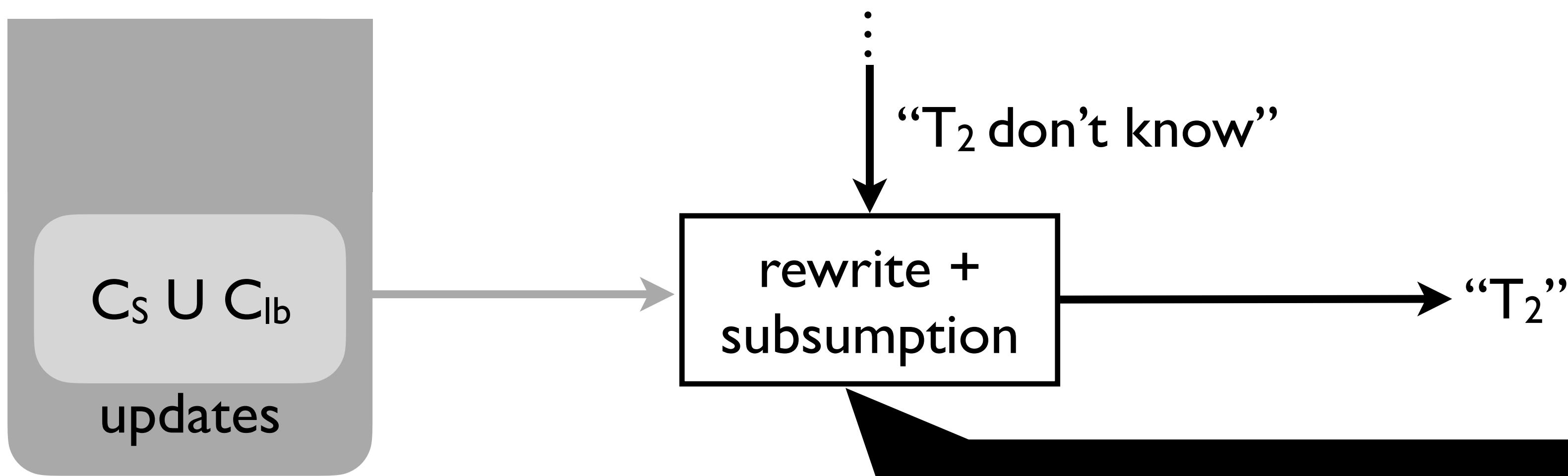
```
panic :- Vt(x,y,p)

Vt(x̄,CS,p̄) :-
R(x̄,CS,p̄),x̄≠Mkt,x̄≠R&D.

Vt(x̄,CS,p̄) :-
R(x̄,CS,p̄),¬Lb(x̄,CS)

Vt(x̄,CS,p̄) :-
R(x̄,CS,p̄),p̄≠7000
```

subsumes

T_2'

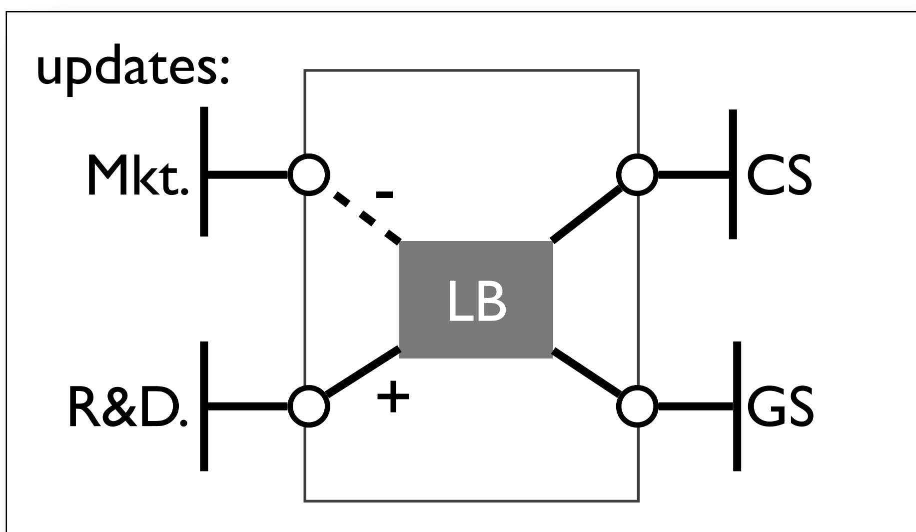
```
/* add (R&D,GS) to LB */
Lb(R&D,GS).
Lb1(x̄,ȳ) :- Lb(x̄,ȳ)

/* delete (Mkt,CS) LB */
Lb2(x̄,ȳ):- Lb1(x̄,ȳ)[x̄≠Mkt]
Lb2(x̄,ȳ) :- Lb1(x̄,ȳ)[ȳ≠CS]

/*panic after updates */
panic :- R(R&D,ȳ,7000),
         ¬Lb2(R&D,ȳ)
```

category (ii) test: using constraints & updates



proposition

given a constraint C and an update U, incorporate U into C by rewriting C to C′: C holds after the update U iff C′ holds before the update



C_{lb}

```
panic :- Vt(x,y,p)

Vt(x̄,CS,p̄) :-
R(x̄,CS,p̄),x̄≠Mkt,x̄≠R&D.

Vt(x̄,CS,p̄) :-
R(x̄,CS,p̄),¬Lb(x̄,CS)

Vt(x̄,CS,p̄) :-
R(x̄,CS,p̄),p̄≠7000
```

subsumes

```
Lb(R&D,GS).
Lb1(x̄,ȳ) :- Lb(x̄,ȳ)

/* delete (Mkt,CS) LB */
Lb2(x̄,ȳ):- Lb1(x̄,ȳ)[x̄≠Mkt]
Lb2(x̄,ȳ) :- Lb1(x̄,ȳ)[ȳ≠CS]

/*panic after updates */
panic :- R(R&D,ȳ,7000),
         ¬Lb2(R&D,ȳ)
```

preliminary result

practical implementation in SQL

- shallow embedding of fauré-log in PostgreSQL + Z3

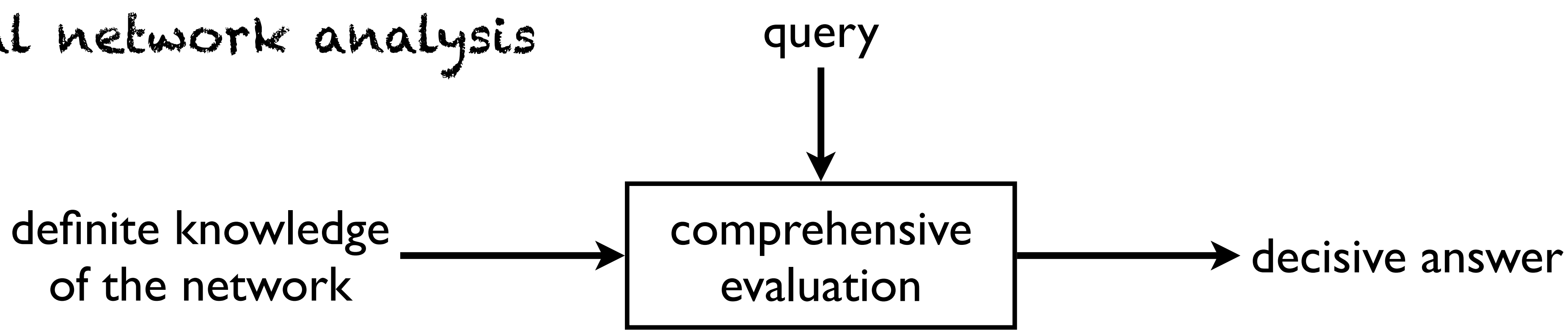
evaluation

- realistic topology (inferred from BGP announcements)
- synthetic link failures
- representative queries
  - q_4-q_5 (all pair-wise reachability), q_6-q_8 (various failure patterns)

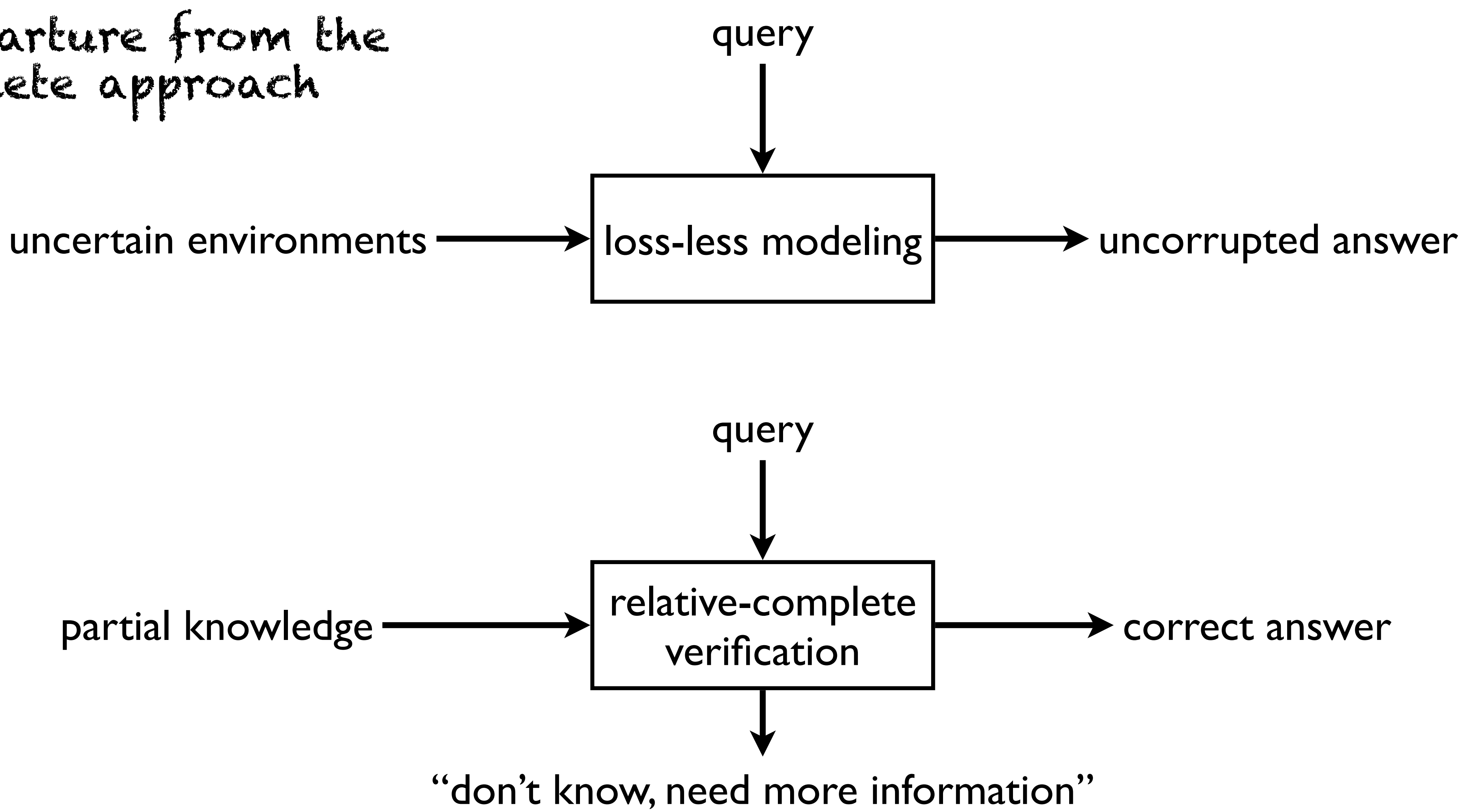
	$q_4 - q_5$	q_6			q_7			q_8		
#prefix	sql	sql	Z3	#tuples	sql	Z3	#tuples	sql	Z3	#tuples
1000	0.625s	0.85s(0.11%)	796.35s	42425	0.08s(22.86%)	0.27s	16	0.15s(1.17%)	12.64s	828
10000	5.75s	8.96s	-	418224	0.27s(7.33%)	3.41s	194	1.8s(1.27%)	137.05s	8706
100000	54.85s	113.48s	-	4435862	1.66s(6.18%)	25.22s	1387	34.67s(1.71%)	1941.04s	86360
922067	816.4s	4169.02s	-	46503247	11.1s(3.71%)	288.17s	16490	267.05s	-	858180

recap — partial analysis

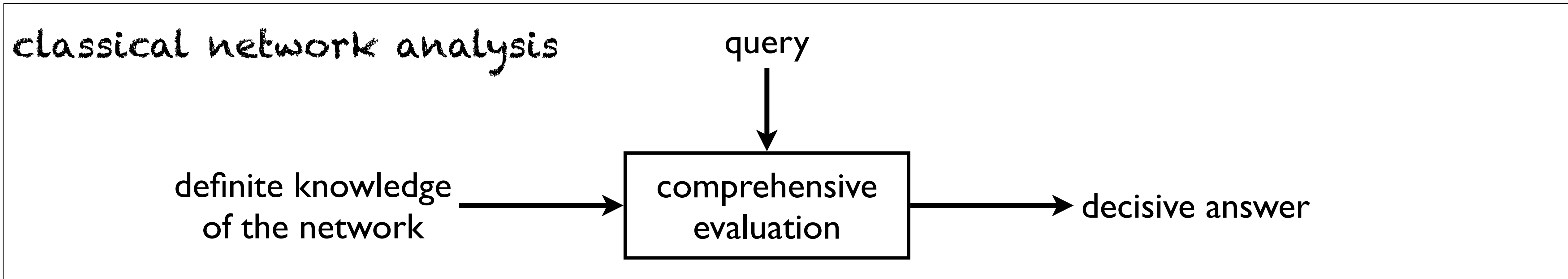
classical network analysis

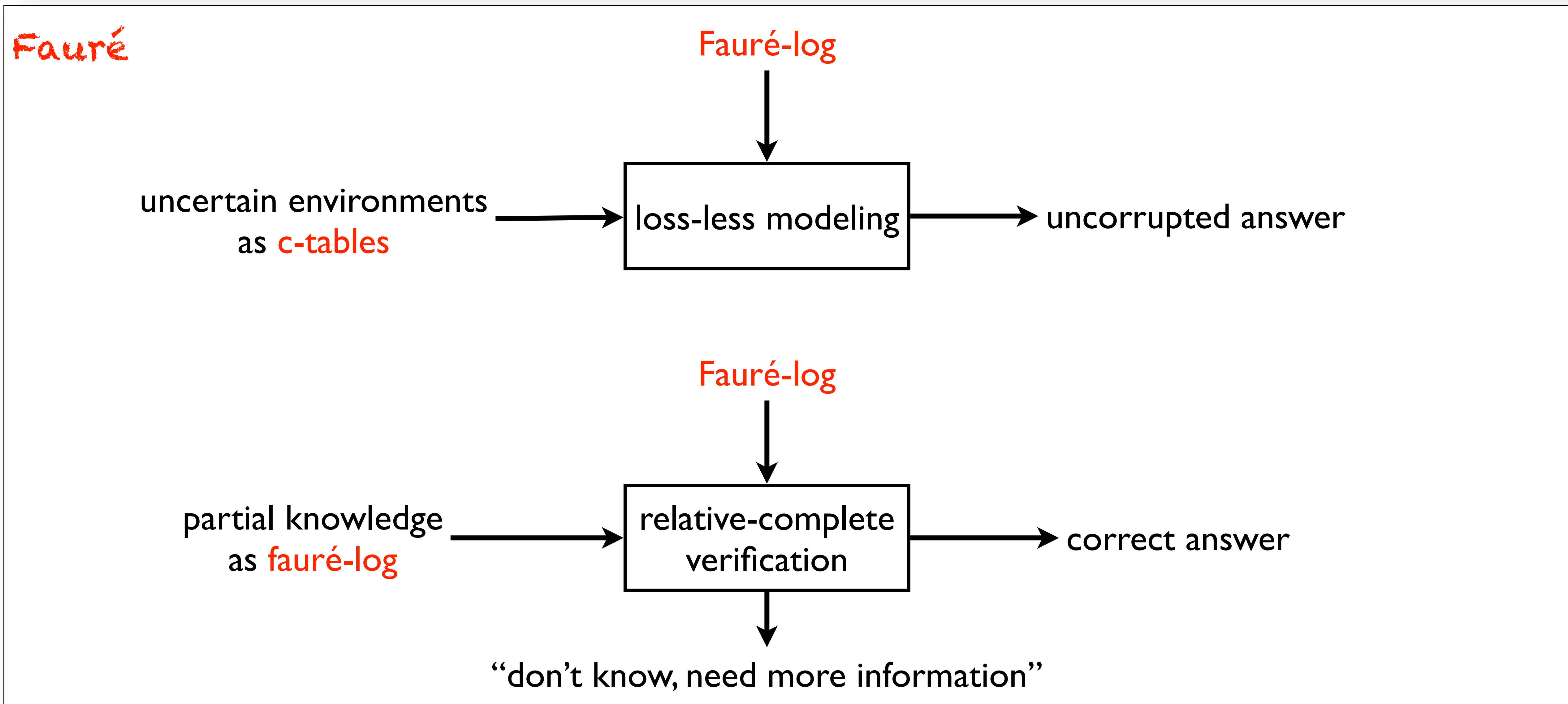


a departure from the complete approach

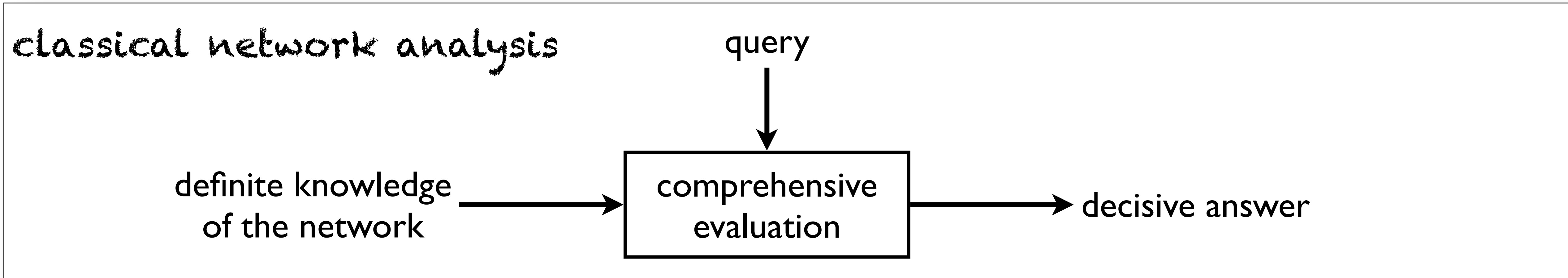


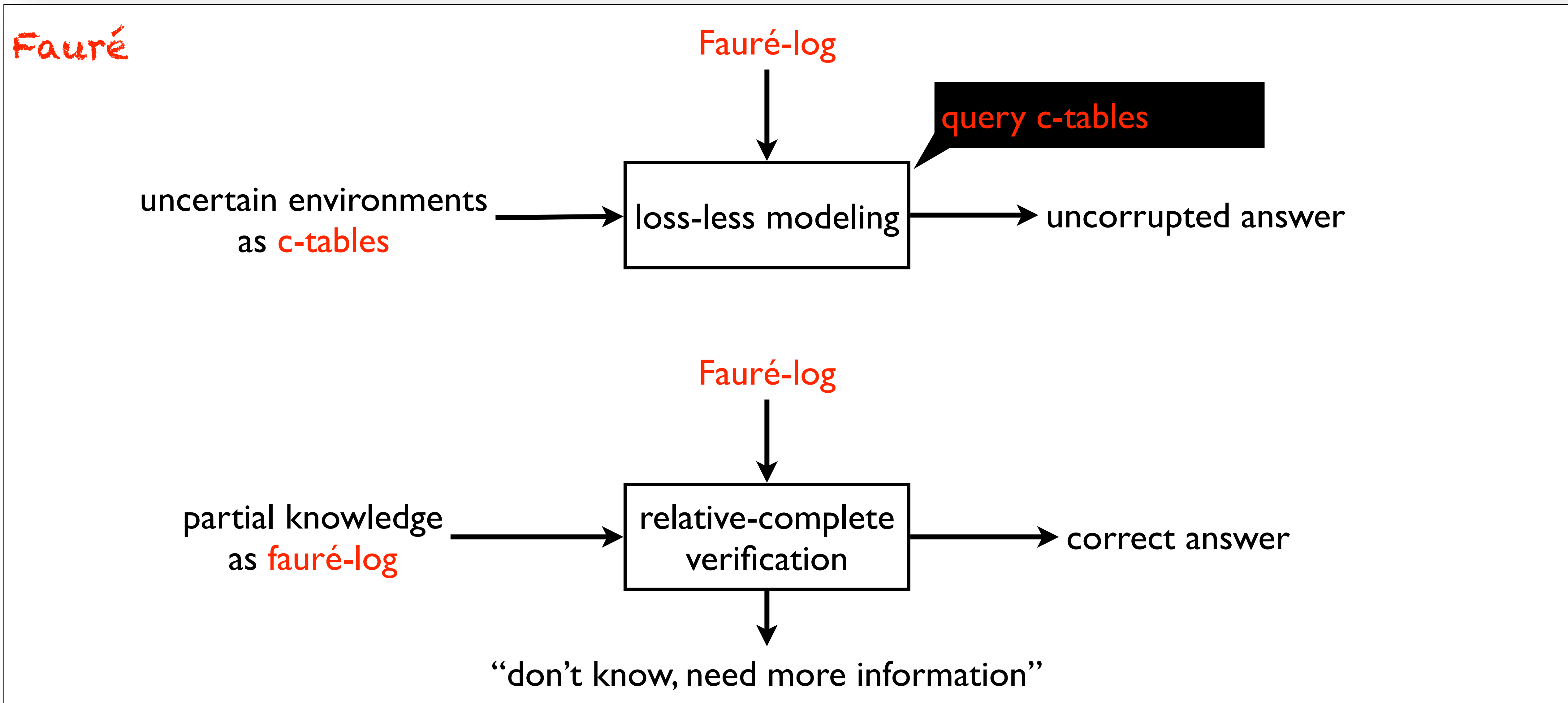
recap — realization



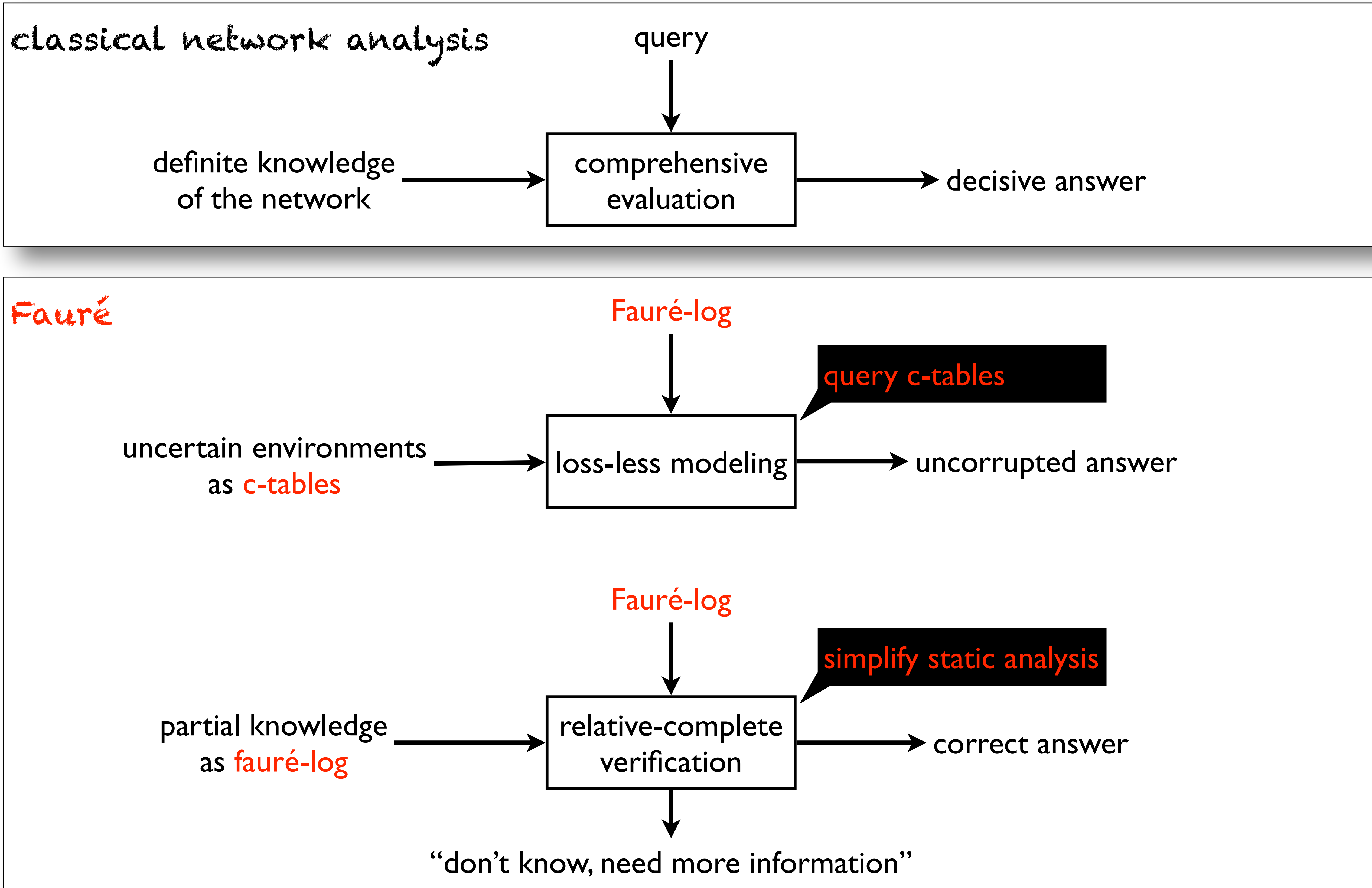


recap — realization





recap — realization



thank you

https://github.com/ravel-net/Faure