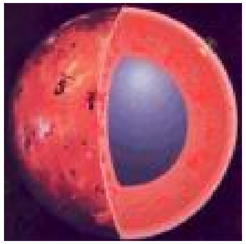
Computing Cores for Data Exchange
New Algorithms and Practical Solutions



Georg Gottlob
TU Wien

Talk Structure

Brief introduction

General core computation (no dependencies)

- Core computation via hypertree decompositions $O(n^{b+3}) \rightarrow O(n^{\underline{b/2}+2})$
- Fixed-Parameter Intractability w.r.t. blocksize b

Computing cores in presence of dependencies **(TGDs & EGDs)**

- A tractable class: Simple TGDs + arbitrary EGDs
- Another tractable class: Full TGDs + arbitrary EGDs
- NP-complete problem variations

Cores

Instance:

{ p(X,Y), p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) }

Logical meaning

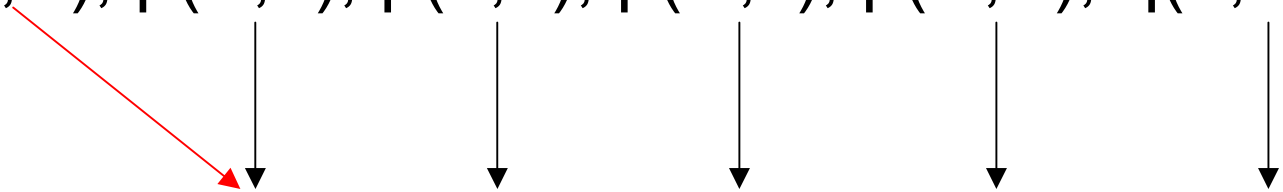
∃ X, Y, U, V:

p(X,Y) **&** p(X,b) **&** p(a,b) **&** p(U,c) **&** p(U,V) **&** q(a,c,d)

Cores

endomorphism h: {Y → b}

I = { p(X,Y), p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) }



{ p(X,Y), p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) }

Cores

endomorphism h: {Y → b}

I = { p(X,Y), p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) }

{ p(X,Y), p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) }

REDUNDANT!

∃X,Y p(X,Y) & P(X,b)

⇑⇓

∃X p(X,b)

Cores

endomorphism h: {Y → b}

I = { p(X,Y), p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) }

{ ~~p(X,Y)~~, p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) }

REDUNDANT!

$\exists X,Y\ p(X,Y)$

$\Uparrow$

$\exists X\ p(X,b)$

Cores

endomorphism h: {Y → b}

I = { p(X,Y), p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) }

⇔

h(I) = { p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) }

Cores

endomorphism h: {Y → b}

I = { p(X,Y), p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) }

⇔

h(I) = { p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) }

X → a

V → c

h(I) can be further reduced by endomorphism g: {X → a, V→ c}

Cores

endomorphism h: {Y → b}

I = { p(X,Y), p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) }

⇔

h(I) = { p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) }

{ p(a,b), p(U,c), q(a,c,d) }

h(I) can be further reduced by endomorphism g: {X → a, V → c}

Cores

$I = \{ p(X,Y), p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) \}$

$\Leftrightarrow$

$h(I) = \{ p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) \}$

$\Leftrightarrow$

$f(I) = g(h(I)) = g \circ h(I) = \{ p(a,b), p(U,c), q(a,c,d) \}$

endomorphism f: {X → a, Y→b, V→ c}

Cores

I = { p(X,Y), p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) }

⇔

h(I) = { p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) }

⇔

f(I)= g(h(I)) = g○h(I)= { p(a,b), p(U,c), q(a,c,d) }

no refinement by endomorphisms possible !

endomorphism f: {X → a, Y→b, V→ c}

Cores

I = { p(X,Y), p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) }

⇔

h(I) = { p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) }

⇔

f(I)= g(h(I)) = g○h(I)= { p(a,b), p(U,c), q(a,c,d) }

Core(I)

unique up to variable-renaming!

endomorphism f: {X → a, Y→b, V→ c}

Blocks

I = { p(X,Y), p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) }

Blocks: Connected components in the variable-graph

Atom-Blocks: corresponding sets of atoms

Blocks

I = { p(X,Y), p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) }

{X,Y}   {U,V}   blocksize(I)=2

Blocks: Connected components in the variable-graph

Atom-Blocks: corresponding sets of atoms

blocksize(I) = size of largest block of I

Computing core(I) is NP-hard in general.

It is tractable for bounded blocksize
[Fagin, Kolaitis, Popa]

We can use a suitable adaptation of
the DC algorithm [G.,Leitsch 85]

Core Computation

$I \; = \; \{ p(X,Y), p(X,b), p(a,b), p(U,c), p(U,V), q(a,c,d) \}$

[Fagin et al]: Compute Core(I) by successively
finding and applying
useful local endomorphisms

useful: at least 1 variable disappears
local: identity, except for 1 block

Can be regarded as a series of query-containment checks
I contains J, I ► J iff ∃ homomorphism h: h(I) ⊆ J

Algorithm CORECOMP

INPUT: An instance I
OUTPUT: Core(I)

1. Identify the blocks of I; B(X) denotes block of X in I;

2. K := I;

3. FOR each X $\in$ var(I) DO
   IF X $\in$ var(K) AND K[B(X)] ► (K – atoms (X,K))
   THEN K := K – atoms(X,K);

4. Output K.

Algorithm CORECOMP

INPUT: An instance I
OUTPUT: Core(I)

1. Identify the blocks of I; B(X) denotes block of X in I;

2. K := I;

3. FOR each X ∈var(I) DO
   IF X ∈ var(K) AND K[B(X)] ►(K – atoms (X,K))
   THEN K := K – atoms(X,K);

4. Output K.

blocksize=b

≤ **b variables**

Essentially |var(I)| block containment tests

Each takes $O(n^{b+2})$ steps. ➔ runtime $O(n^{b+3})$

Algorithm CORECOMP

INPUT: An instance I
OUTPUT: Core(I)

1. Identify the blocks of I; B(X) denotes block of X in I;

2. K := I;

3. FOR each X $\in$ var(I) DO
   IF X $\in$ var(K) AND K[B(X)] ► (K – atoms (X,K))
   THEN K := K – atoms(X,K);

4. Output K.

blocksize=b

$\leq$ **b variables**

Essentially |var(I)| block containment tests

Each takes $O(n^{b+2})$ steps. ➔ runtime $O(n^{b+3})$

we improve this bound

THEOREM:

Let Q and Q´ be conjunctive queries.

Checking Q ► Q´ can be done in time

$O(|Q|\, n^{b/2+1})$ where $b = |var(Q)|$

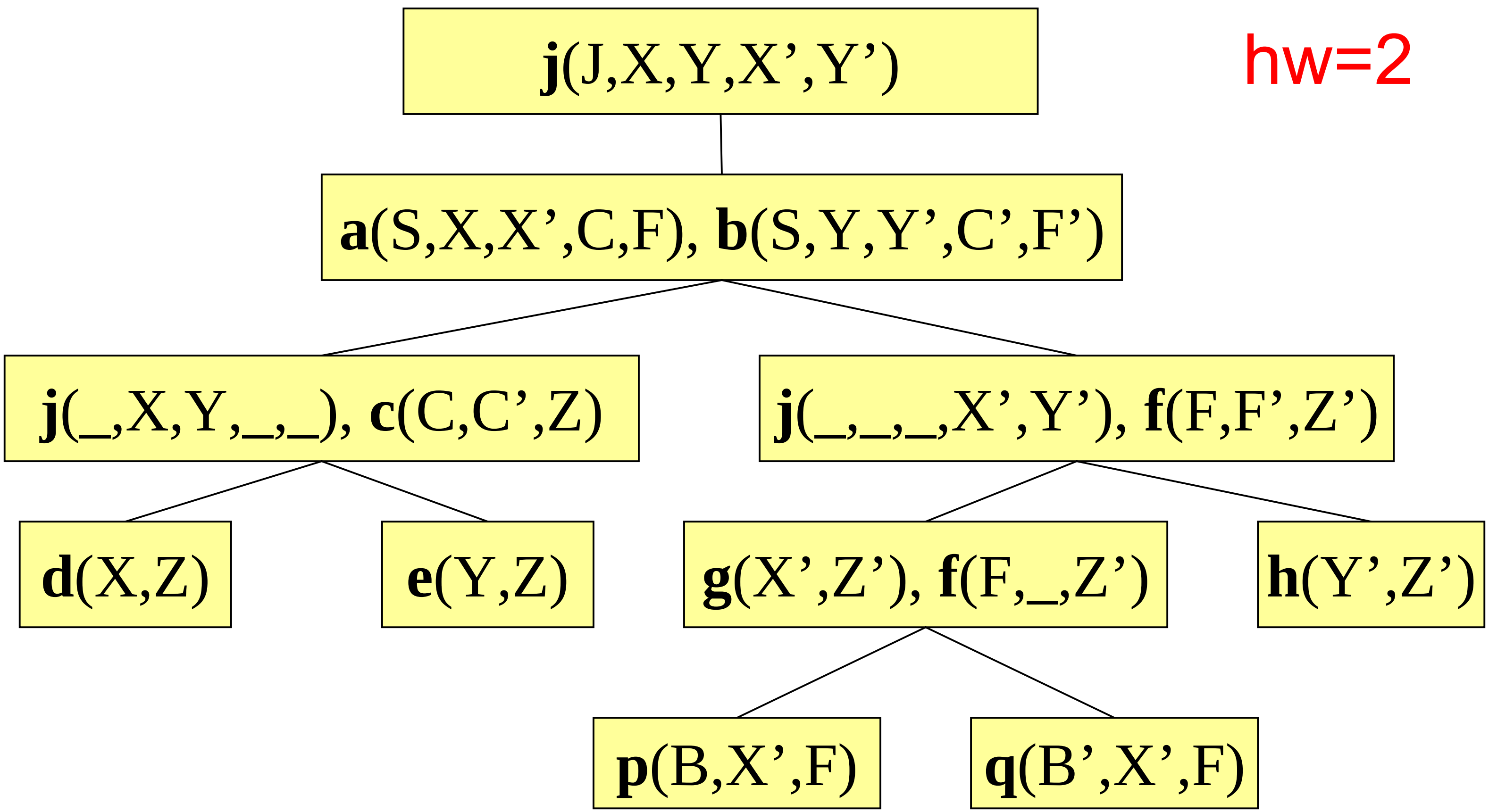
It is well-known that checking whether Q ► Q´ is equivalent to evaluate the Boolean query Q over can(Q´)

Proof (idea).

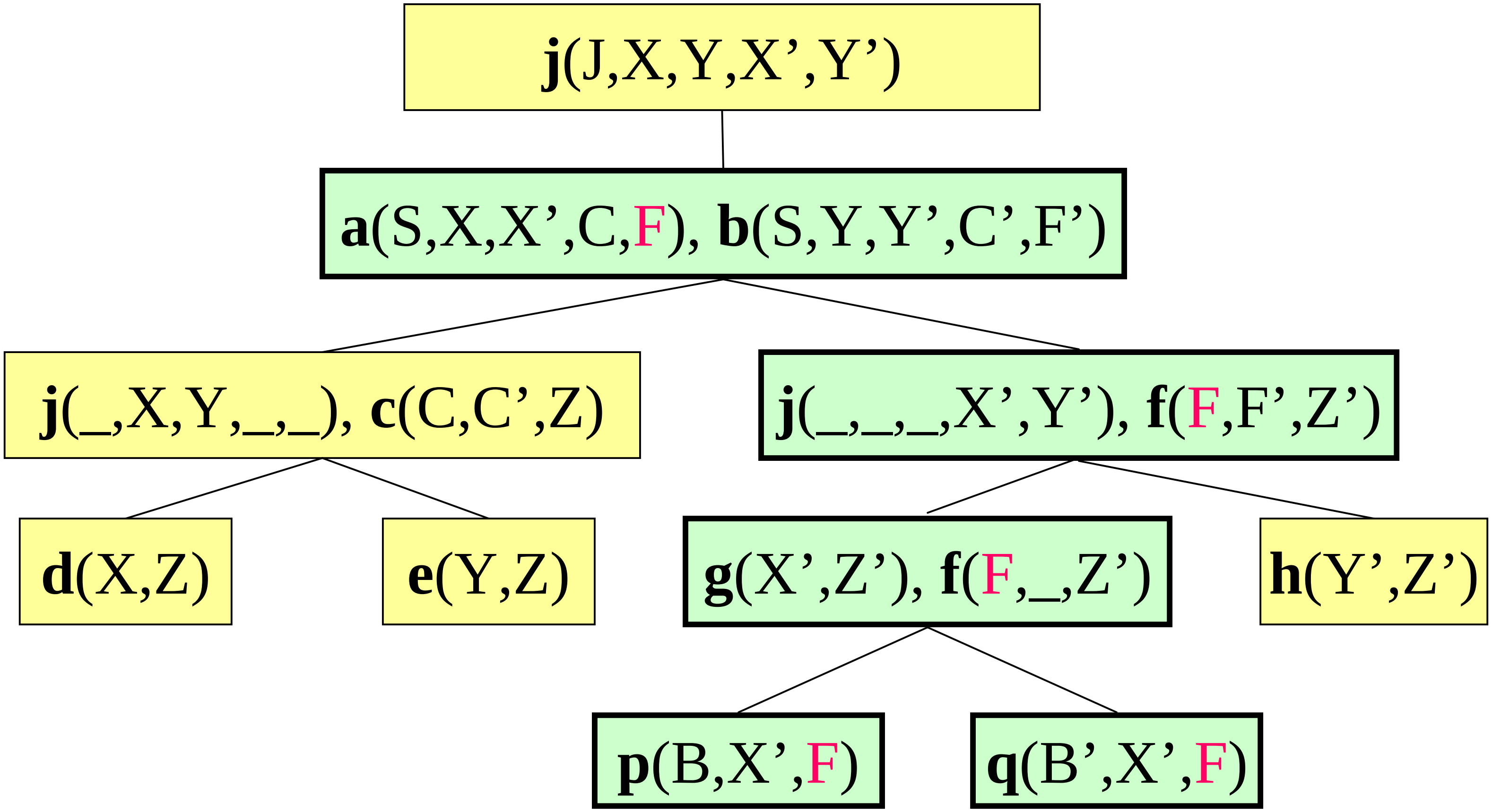
Use hypertree decompositions.

[G., Leone, Scarcello 98]

$$ans \leftarrow a(S,X,X',C,F) \wedge b(S,Y,Y',C',F') \wedge c(C,C',Z) \wedge d(X,Z) \wedge$$
$$e(Y,Z) \wedge f(F,F',Z') \wedge g(X',Z') \wedge h(Y',Z') \wedge$$
$$j(J,X,Y,X',Y') \wedge p(B,X',F) \wedge q(B',X',F)$$



connectedness condition



Proof (idea). Use hypertree decompositions. [G., Leone, Scarcello 98]

We show that each query Q with b variables has hypertree-width ≤ b/2 + 1

It can be transformed into an equivalent acyclic query and evaluated in time $O(n^{b/2+c})$ using Yannakakis´ algorithm for Boolean acyclic queries (i.e. the bottom-up phase only).

Proof (continued).

q(x,y,z),s(z,r), q(u,v,w), s(t,u), s(r,w),q(w,s,t), q(t,r,y),s(r,z),q(r,y,z)

Add atoms with 2 new variables as long as Possible.

Proof (continued).

q(x,y,z),s(z,r), q(u,v,w), s(t,u), s(r,w),q(w,s,t), q(t,r,y),s(r,x),q(r,y,z)

Add atoms with 2 new variables as long as Possible.

Proof (continued).

$q(x,y,z)$, $s(z,r)$, $q(u,v,w)$, $s(t,u)$, $s(r,w)$, $q(w,s,t)$, $q(t,r,y)$, $s(r,x)$, $q(r,y,z)$

Add atoms with 2 new variables as long as Possible.

Proof (continued).

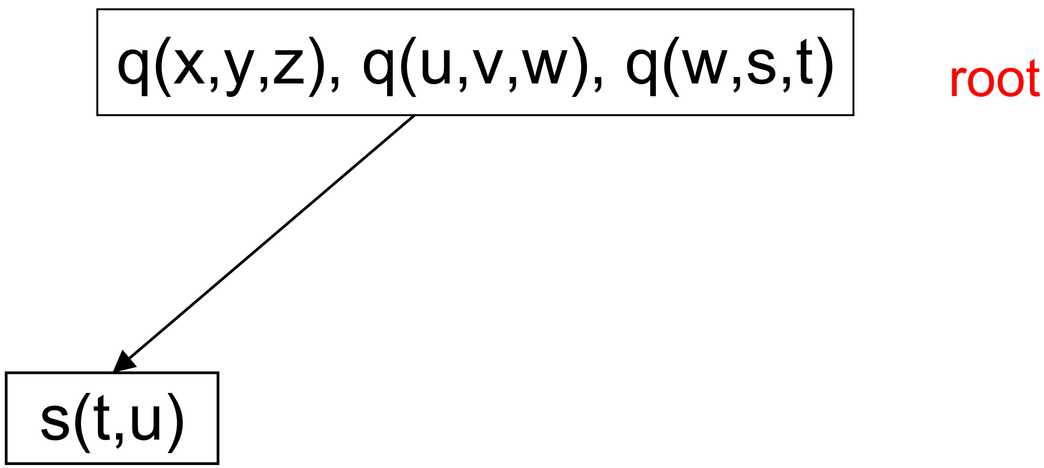
q(x,y,z),s(z,r), q(u,v,w), s(t,u), s(r,w),q(w,s,t), q(t,r,y),s(r,x),q(r,y,z)

q(x,y,z), q(u,v,w), q(w,s,t)

root

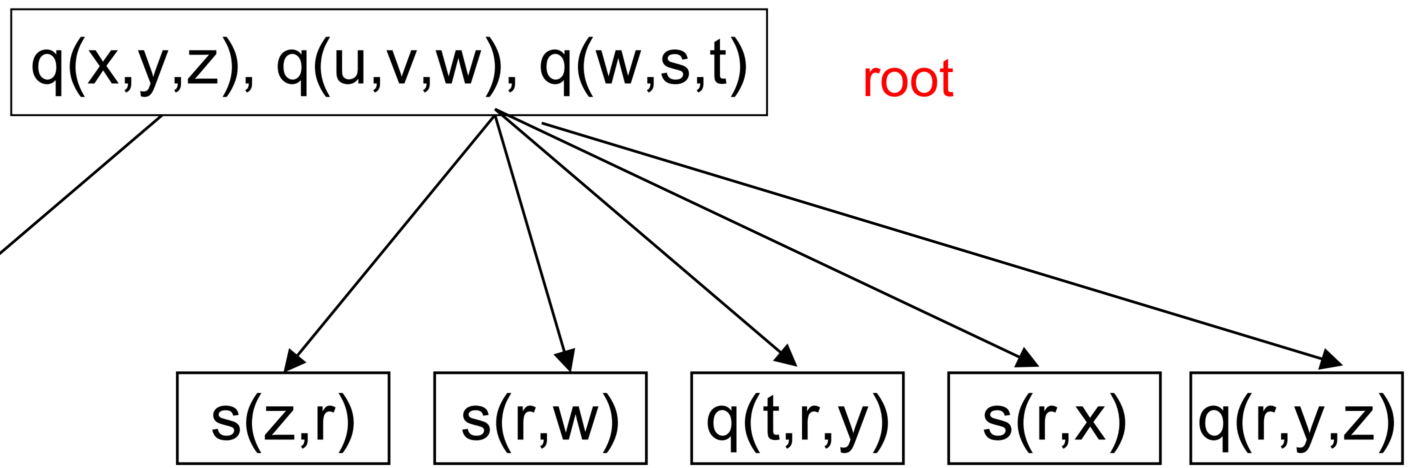
Proof (continued).

q(x,y,z),s(z,r), q(u,v,w), s(t,u), s(r,w),q(w,s,t), q(t,r,y),s(r,x),q(r,y,z)



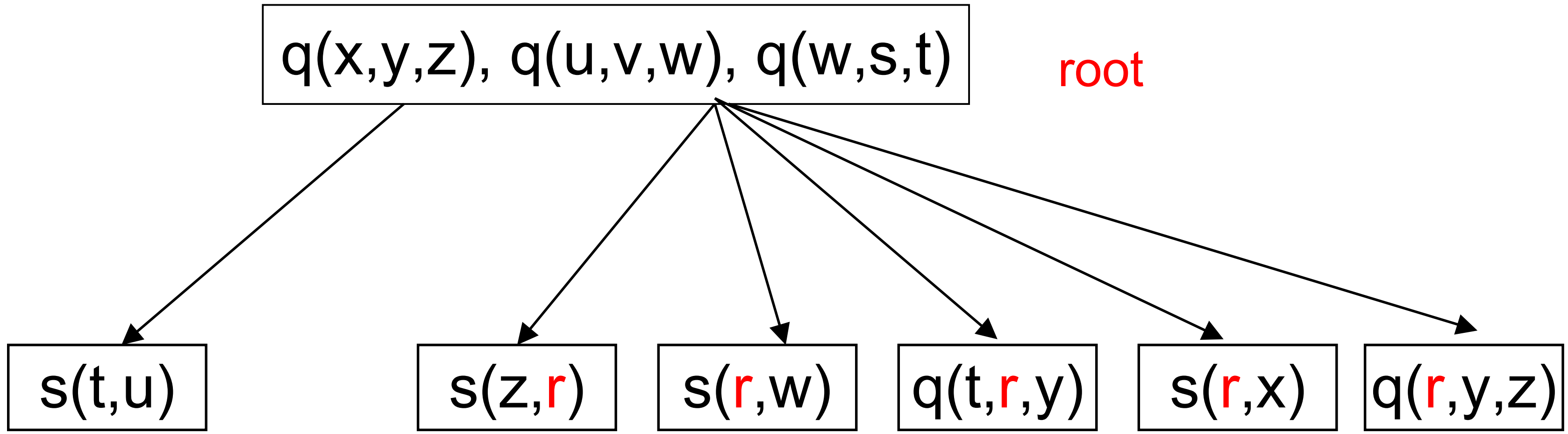
Proof (continued).

q(x,y,z),s(z,r), q(u,v,w), s(t,u), s(r,w),q(w,s,t), q(t,r,y),s(r,x),q(r,y,z)



Proof (continued).

q(x,y,z),s(z,r), q(u,v,w), s(t,u), s(r,w),q(w,s,t), q(t,r,y),s(r,x),q(r,y,z)

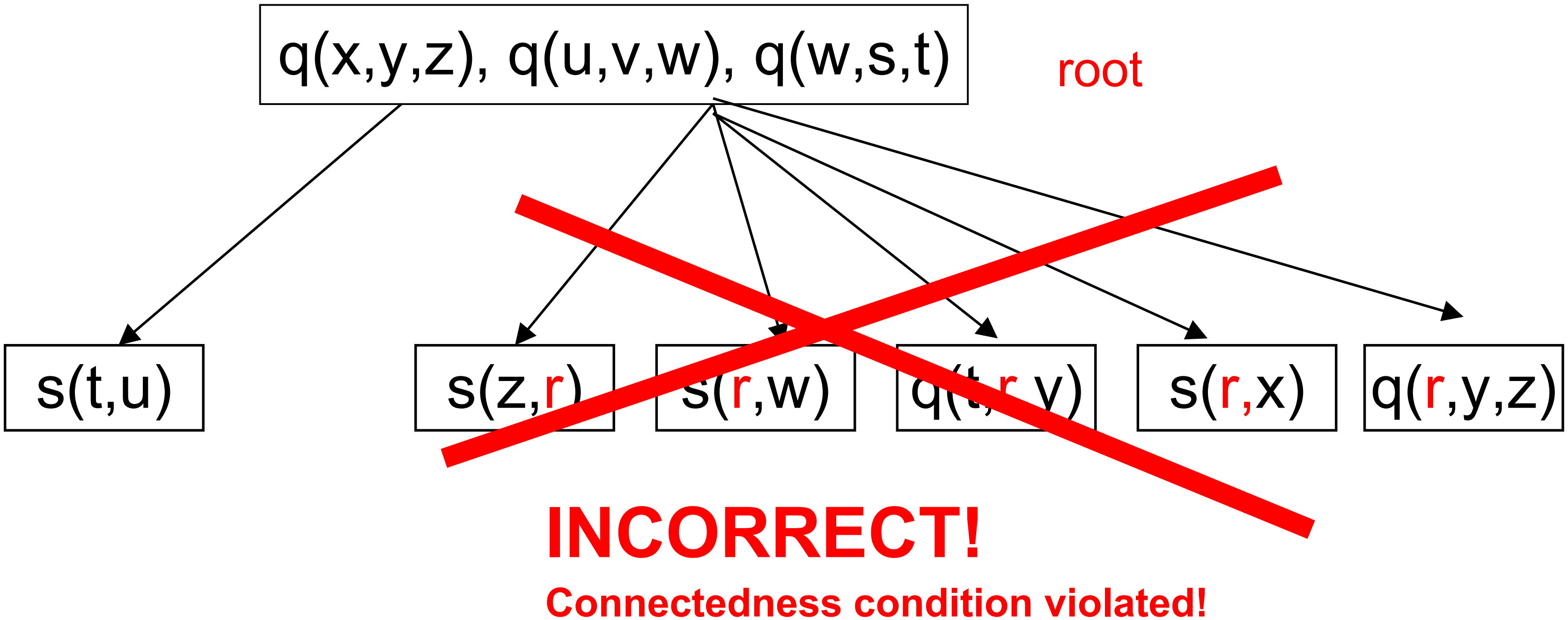


INCORRECT!

Connectedness condition violated!

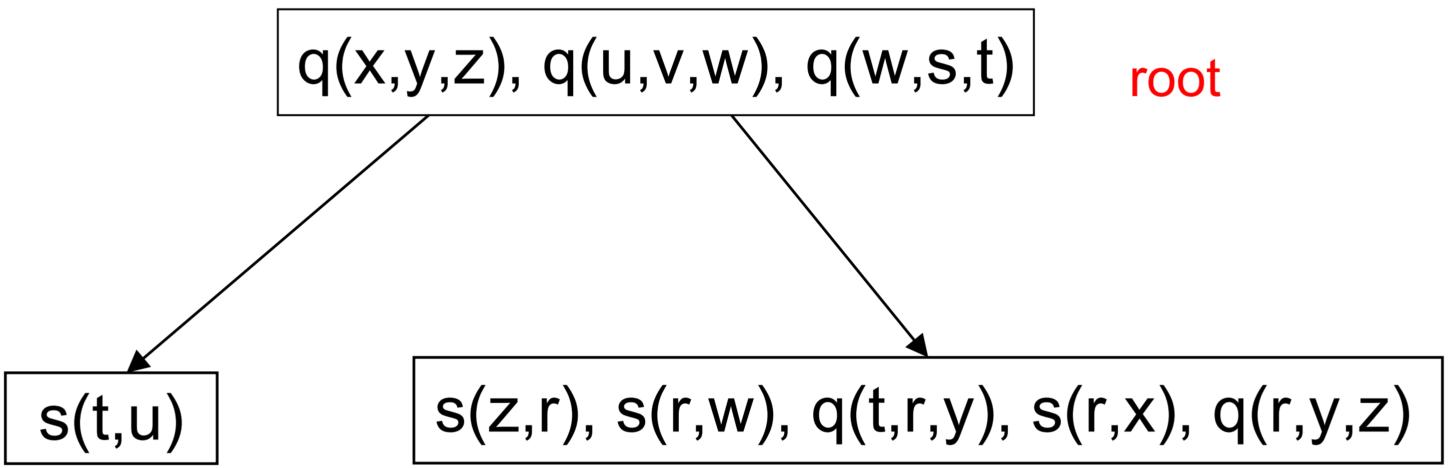
Proof (continued).

q(x,y,z),s(z,r), q(u,v,w), s(t,u), s(r,w),q(w,s,t), q(t,r,y),s(r,x),q(r,y,z)



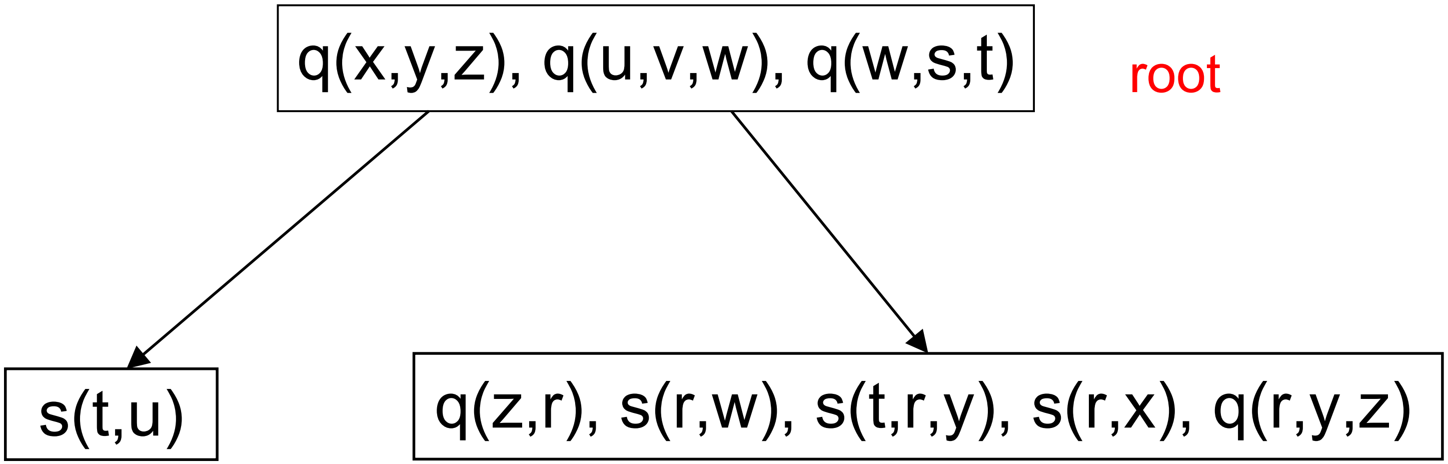
Proof (continued).

q(x,y,z),s(z,r), q(u,v,w), s(t,u), s(r,w),q(w,s,t), q(t,r,y),s(r,x),q(r,y,z)



Proof (continued).

q(x,y,z),s(z,r), q(u,v,w), s(t,u), s(r,w),q(w,s,t), q(t,r,y),s(r,x),q(r,y,z)

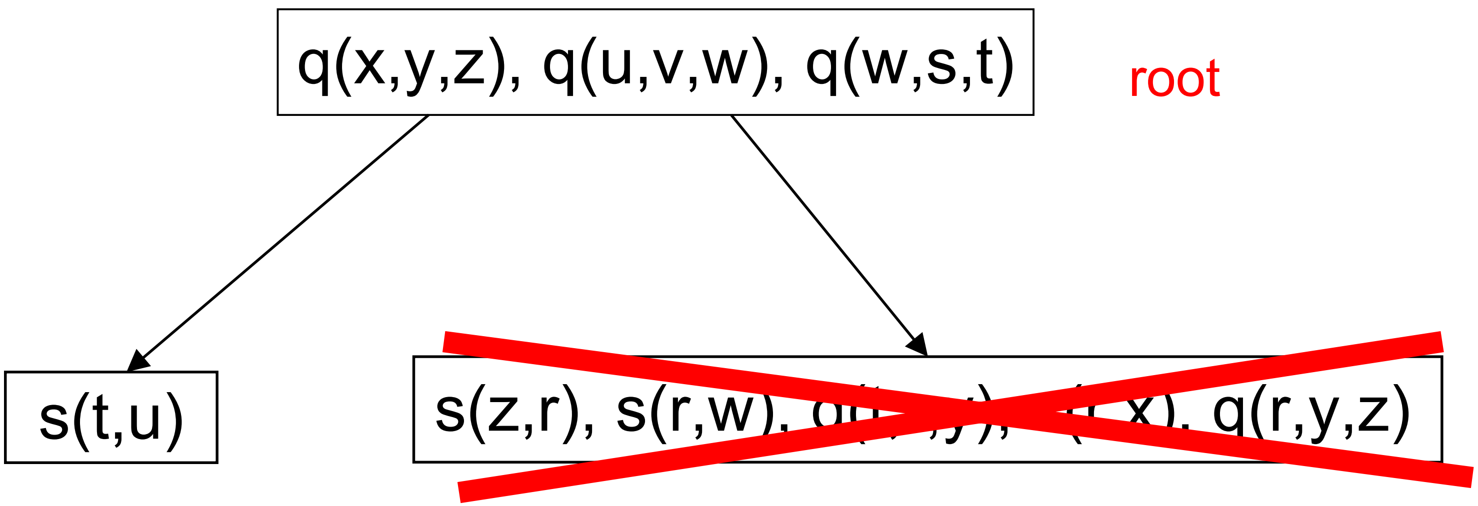


Too large!

Width would increase with each additional r-atom

Proof (continued).

q(x,y,z),s(z,r), q(u,v,w), s(t,u), s(r,w),q(w,s,t), q(t,r,y),s(r,x),q(r,y,z)



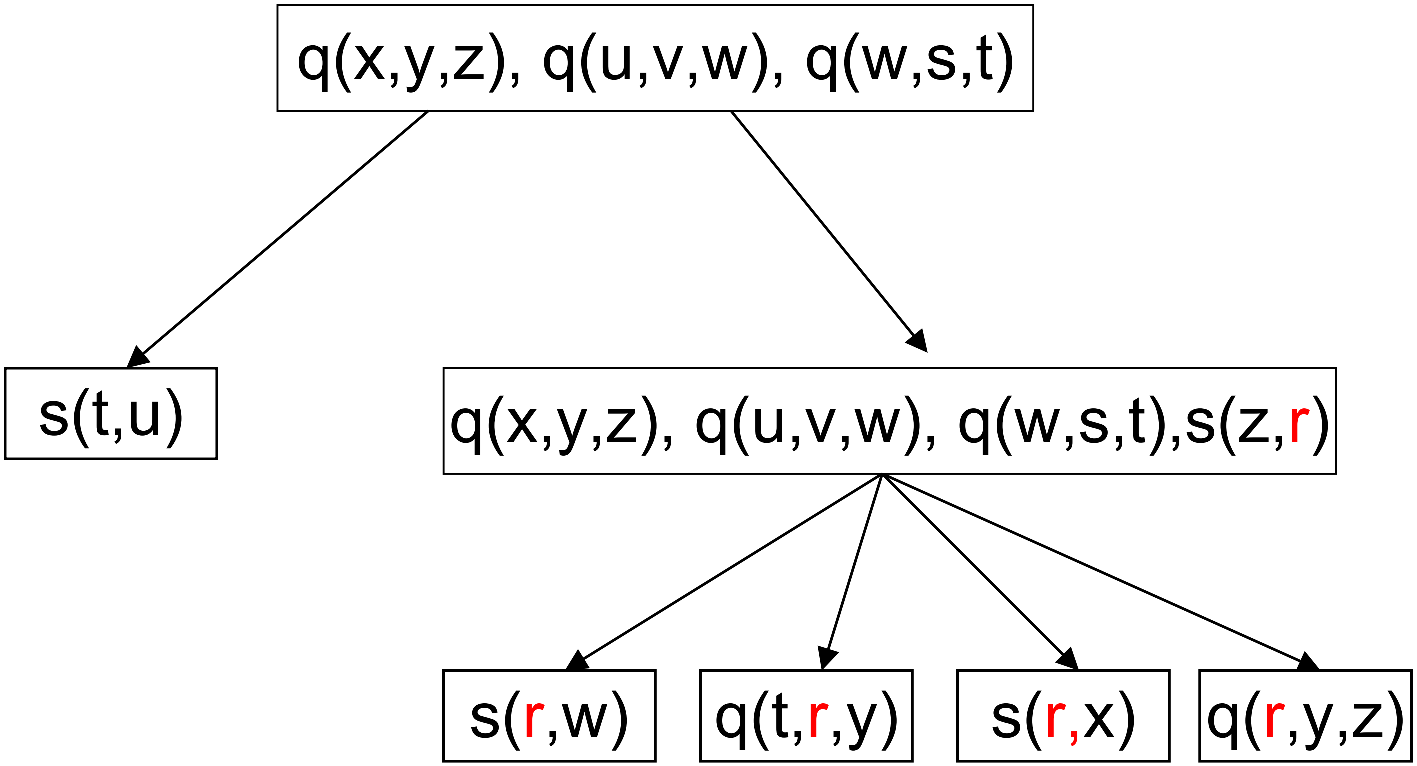
Too large!

Width would increase with each additional r-atom

Proof (continued).

q(x,y,z),s(z,r), q(u,v,w), s(t,u), s(r,w),q(w,s,t), q(t,r,y),s(r,x),q(r,y,z)

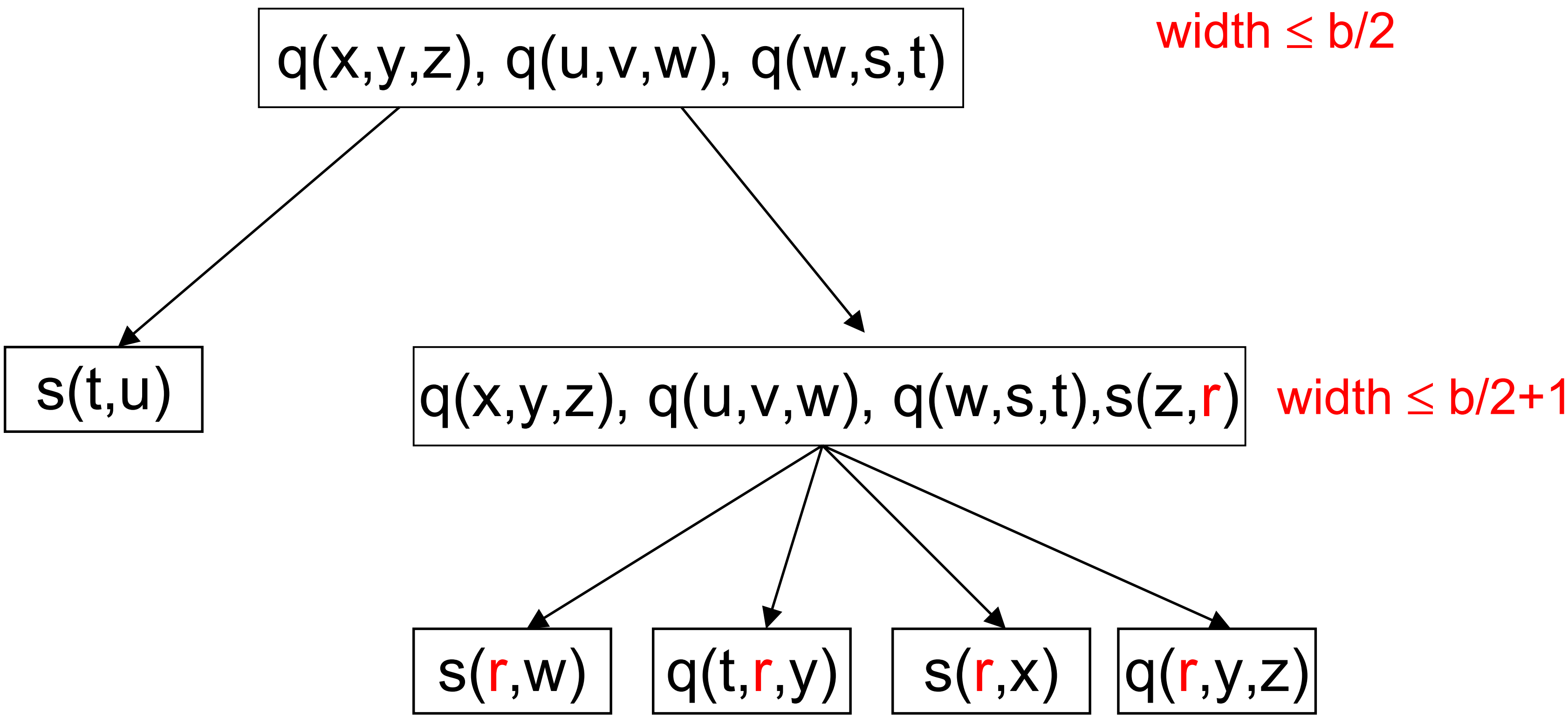
The solution:



Proof (continued).

q(x,y,z),s(z,r), q(u,v,w), s(t,u), s(r,w),q(w,s,t), q(t,r,y),s(r,x),q(r,y,z)

The solution:

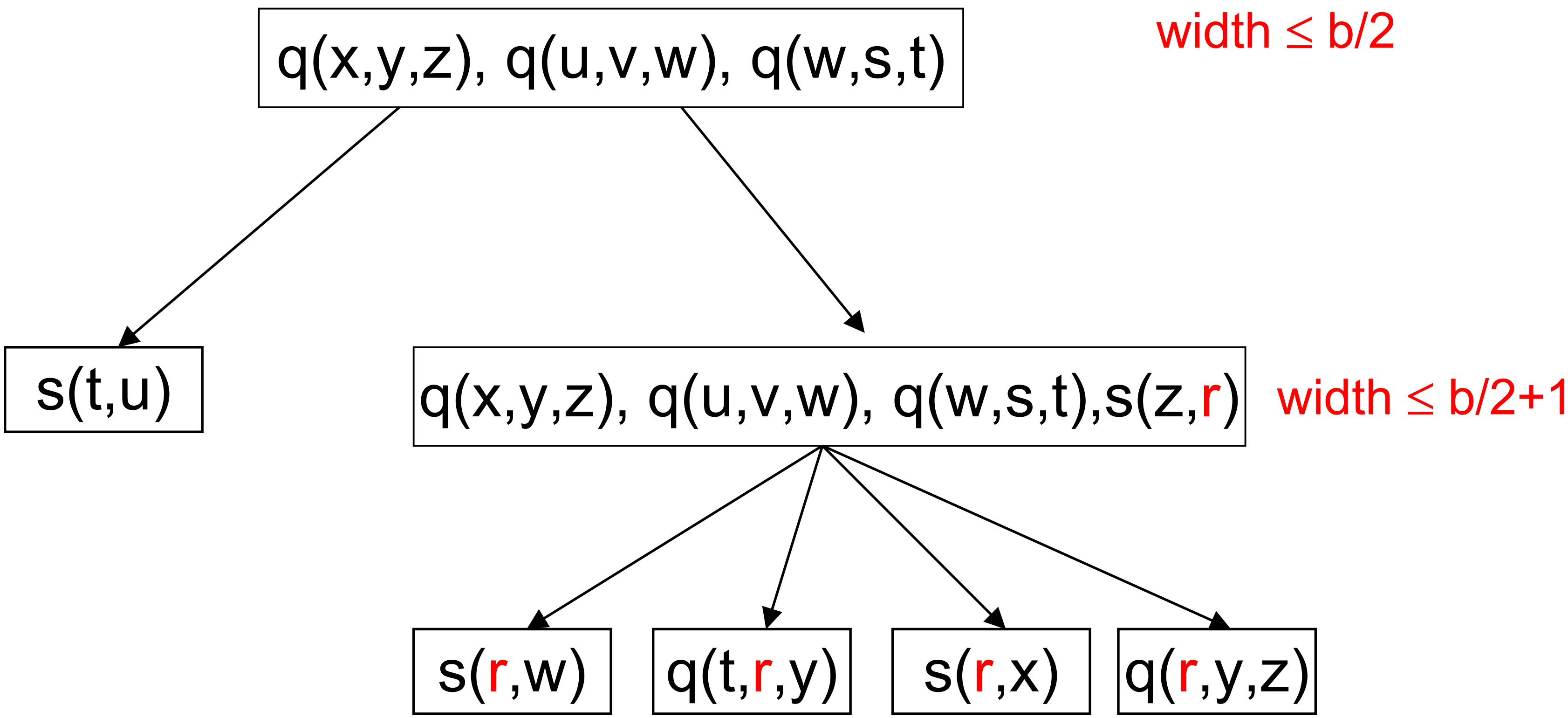


q.e.d

Proof (continued).

q(x,y,z),s(z,r), q(u,v,w), s(t,u), s(r,w),q(w,s,t), q(t,r,y),s(r,x),q(r,y,z)

The solution:



Note: Treewidth instead does not help!

q.e.d

Corollary:

Bound for CORE computation: $O(n^{b/2+2})$.

Question: Can we eliminate the parameter b in the exponent?

Answer: Most likely not !

THEOREM:

Core computation is fixed-parameter intractable w.r.t. the blocksize b.

Dependencies

Tuple generating dependencies   TGDs:

$$p(X,Y)\ \&\ q(Y,Z) \rightarrow \exists\, U,V\ \ r(X,U)\ \&\ p(Z,V)$$

Dependencies

Tuple generating dependencies   TGDs:

$$p(X,Y) \;\&\; q(Y,Z) \rightarrow \exists\, U,V \;\; r(X,U) \;\&\; p(Z,V)$$

Simple TGD: $r(X,Y,Z) \rightarrow \exists\, U,V \;\; s(X,Y,U,V) \;\&\; p(X,Z,V)$

Dependencies

Tuple generating dependencies TGDs:

$$p(X,Y) \,\&\, q(Y,Z) \rightarrow \exists\, U,V\;\; r(X,U) \,\&\, p(Z,V)$$

Simple TGD: $r(X,Y,Z) \rightarrow \exists\, U,V\;\; s(X,Y,U,V) \,\&\, p(X,Z,V)$

Note: Inclusion dependencies are simple TGDs!

Schemas: R(A,B) , S(C,D,E)

IND: $R[A] \subseteq S[C]$

$$r(X,Y) \rightarrow \exists\, U,V\;\; s(X,U,V)$$

Dependencies

Tuple generating dependencies   TGDs:

$$p(X,Y) \,\&\, q(Y,Z) \rightarrow \exists\, U,V \;\; r(X,U) \,\&\, p(Z,V)$$

Simple TGD: $r(X,Y,Z) \rightarrow \exists\, U,V \;\; s(X,Y,U,V) \,\&\, p(X,Z,V)$

Note: INDs are simple TGDs

Weakly acyclic TGDs   (defined for sets of TGDs)

Dependencies

Tuple generating dependencies   TGDs:

$$p(X,Y) \,\&\, q(Y,Z) \rightarrow \exists\, U,V \;\; r(X,U) \,\&\, p(Z,V)$$

Simple TGD: $r(X,Y,Z) \rightarrow \exists\, U,V \;\; s(X,Y,U,V) \,\&\, p(X,Z,V)$

Note: INDs are simple TGDs

Weakly acyclic TGDs   (defined for sets of TGDs)

Full TGDs: no $\exists$ quantifiers e.g. $p(X,Y) \,\&\, p(Y,Z) \rightarrow p(X,Z)$

Dependencies

Tuple generating dependencies TGDs:

$$p(X,Y) \& q(Y,Z) \rightarrow \exists\, U,V\ \ r(X,U) \& p(Z,V)$$

Simple TGD: $r(X,Y,Z) \rightarrow \exists\, U,V\ \ s(X,Y,U,V) \& p(X,Z,V)$

Note: INDs are simple TGDs

Weakly acyclic TGDs (defined for sets of TGDs)

Full TGDs: no ∃ quantifiers e.g. $p(X,Y) \& p(Y,Z) \rightarrow p(X,Z)$

Equality generating dependencies EGDs:

$$p(X,Y) \& p(X,Z) \rightarrow Y=Z$$

Dependencies

Tuple generating dependencies   TGDs:

$$p(X,Y) \,\&\, q(Y,Z) \rightarrow \exists\, U,V \;\; r(X,U) \,\&\, p(Z,V)$$

Simple TGD: $r(X,Y,Z) \rightarrow \exists\, U,V \;\; s(X,Y,U,V) \,\&\, p(X,Z,V)$

Note: INDs are simple TGDs

Weakly acyclic TGDs   (defined for sets of TGDs)

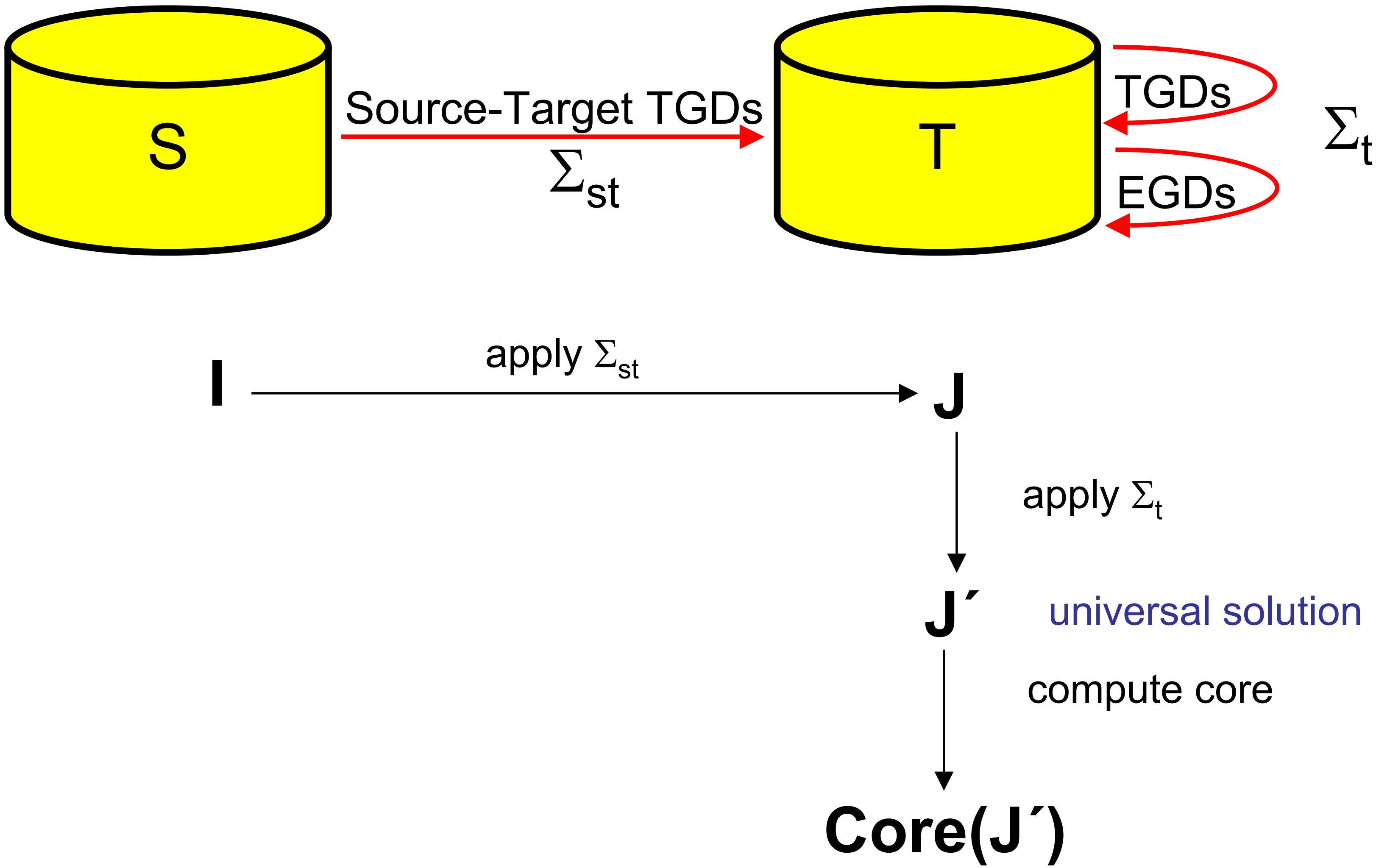
Full TGDs: no ∃ quantifiers e.g. $p(X,Y) \,\&\, p(Y,Z) \rightarrow p(X,Z)$

Equality generating dependencies   EGDs:

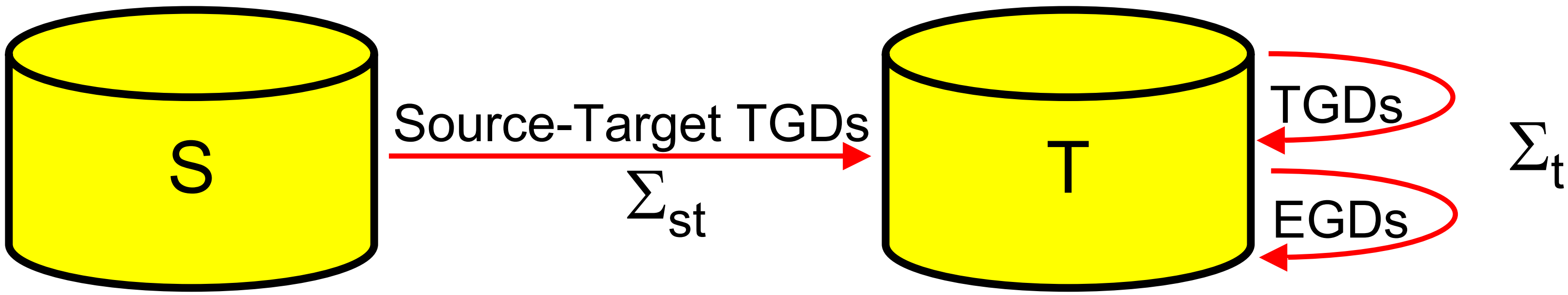
$$p(X,Y) \,\&\, p(X,Z) \rightarrow Y=Z$$

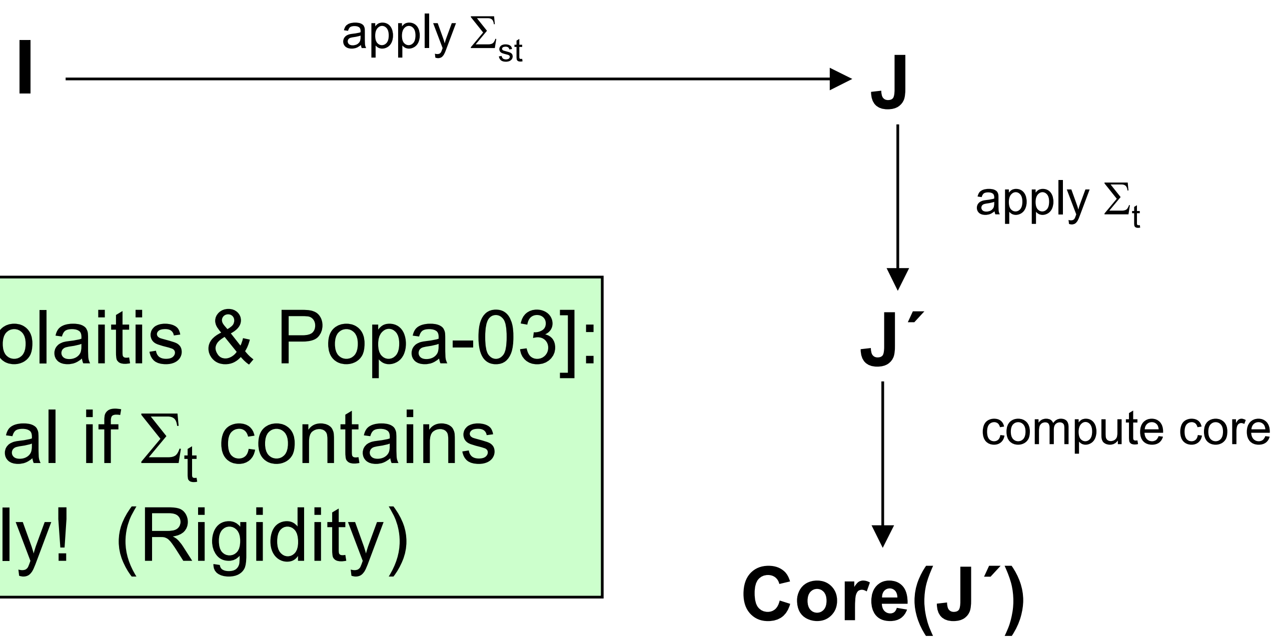
Note: FDs are EGDs

Data Exchange



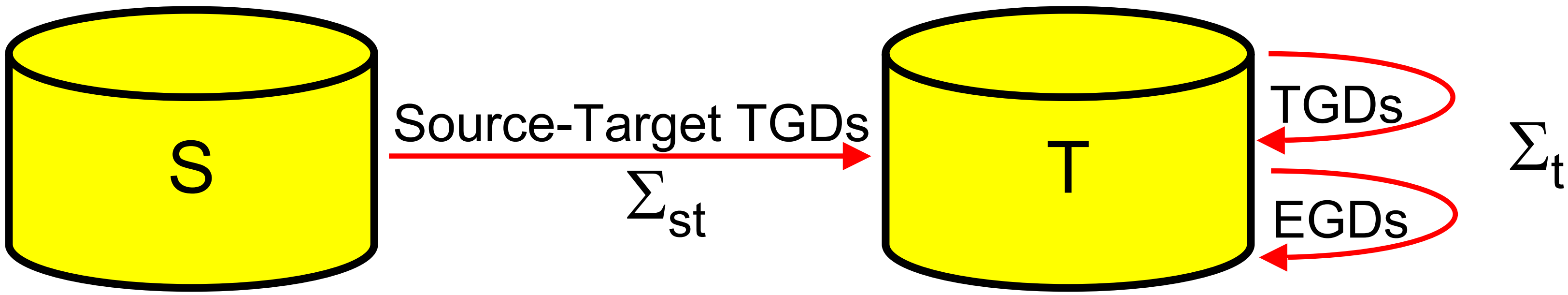
Data Exchange

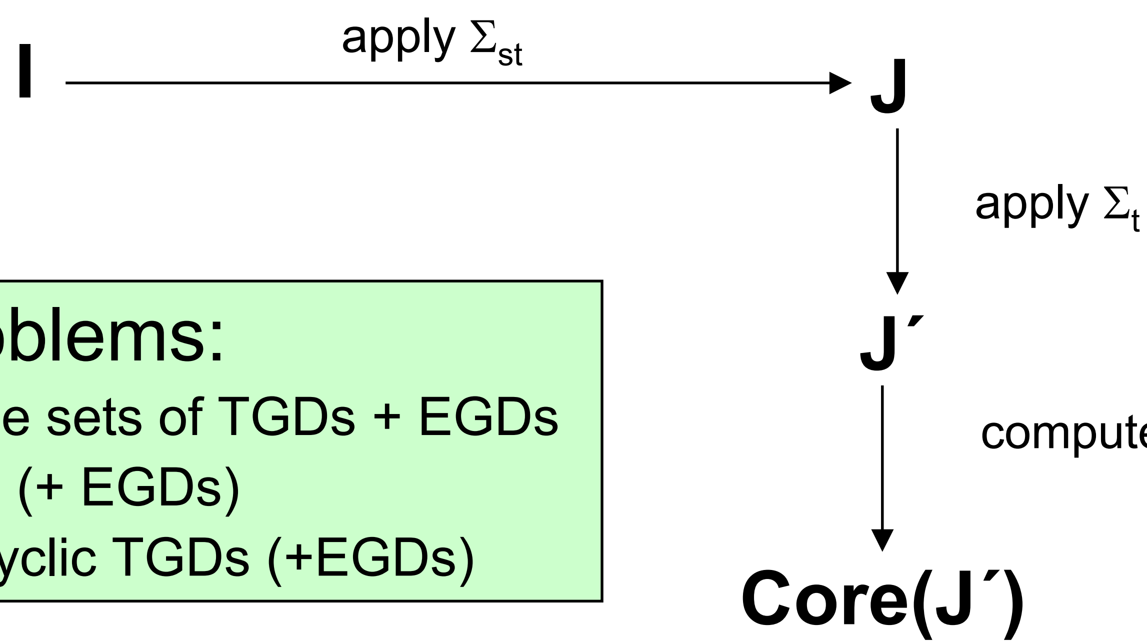




[Fagin, Kolaitis & Popa-03]:
Polynomial if Σ_t contains EGDs only! (Rigidity)

Data Exchange





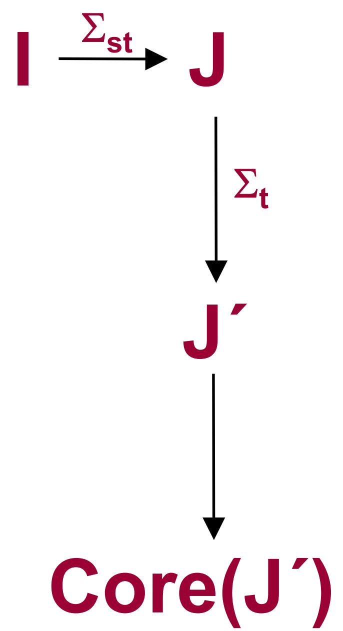
Open problems:

- →Reasonable sets of TGDs + EGDs
- →Full TGDs (+ EGDs)
- →Weakly acyclic TGDs (+EGDs)

Tractability Result 1

THEOREM: Data exchange with EGDs +
weakly acyclic simple TGDs is tractable.

PROOF IDEA:



- J has bounded hypertree width (hw)
- simple TGDs preserve hw
  - ➔ J' has bounded hw
- Use [Fagin et. al.]'s rigidity argument for dealing with EGDs.

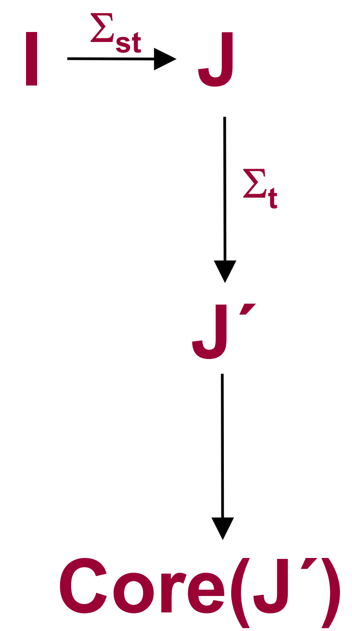
Tractability Result 1

THEOREM: Data exchange with EGDs +
weakly acyclic simple TGDs is tractable.

PROOF IDEA:

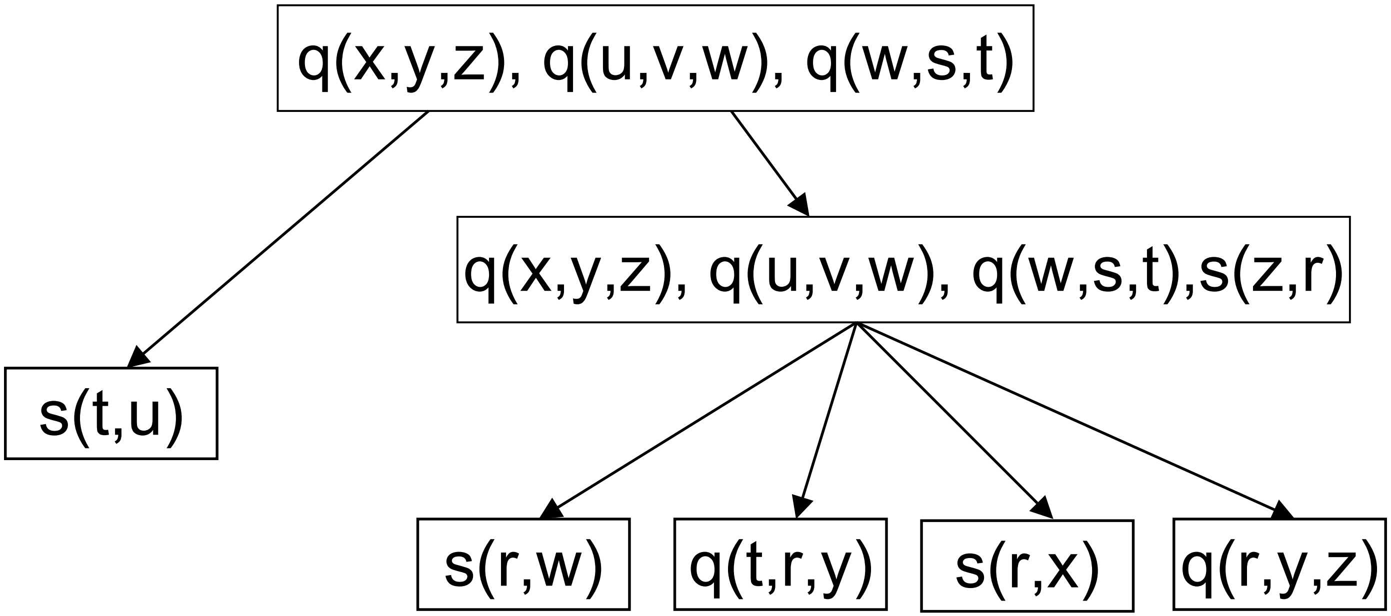
- J has bounded hypertree width (hw)
- <u>simple TGDs preserve hw</u>

  ➔ J' has bounded hw
- Use [Fagin et. al.]'s rigidity argument for dealing with EGDs.



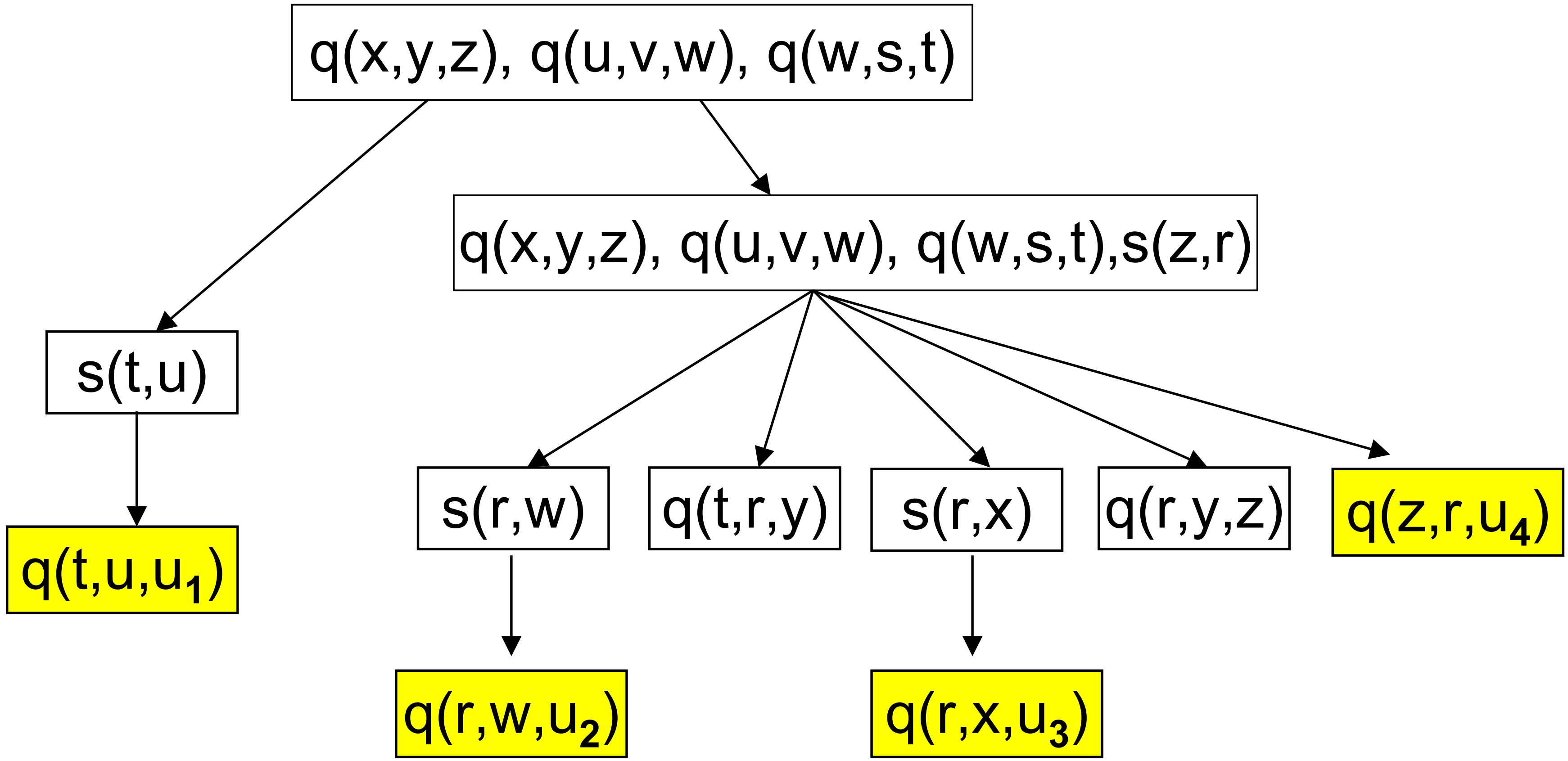
Proof Sketch (continued).

simple TGDs preserve hw $s(x_1,x_2) \rightarrow \exists x_3\ q(x_1,x_2,x_3)$



Proof Sketch (continued).

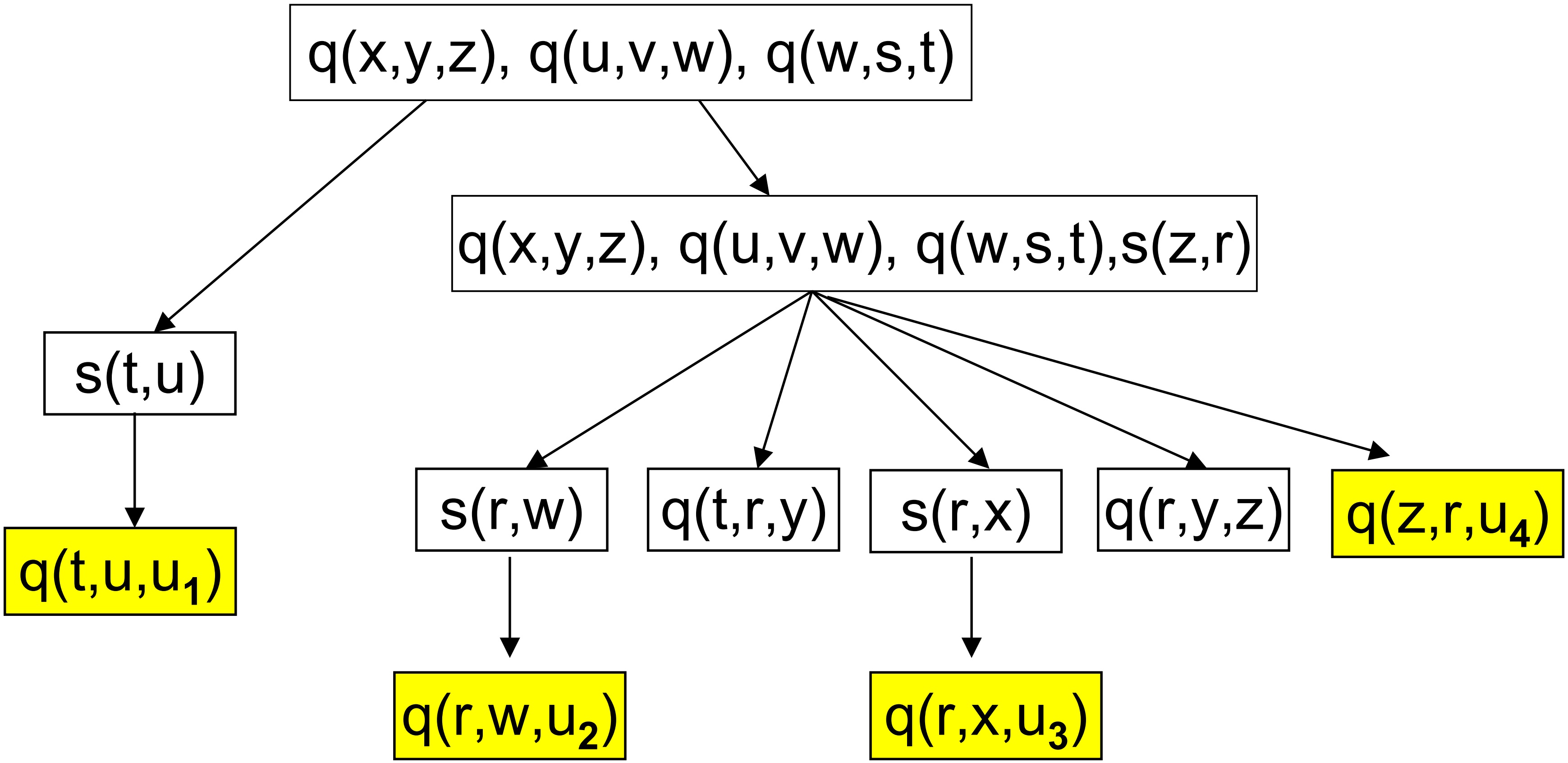
<u>simple TGDs preserve hw</u> $s(x_1,x_2) \rightarrow \exists x_3\ q(x_1,x_2,x_3)$



Proof Sketch (continued).

simple TGDs preserve hw $s(x_1,x_2) \rightarrow \exists x_3\, q(x_1,x_2,x_3)$

Variables from different blocks are never mixed!

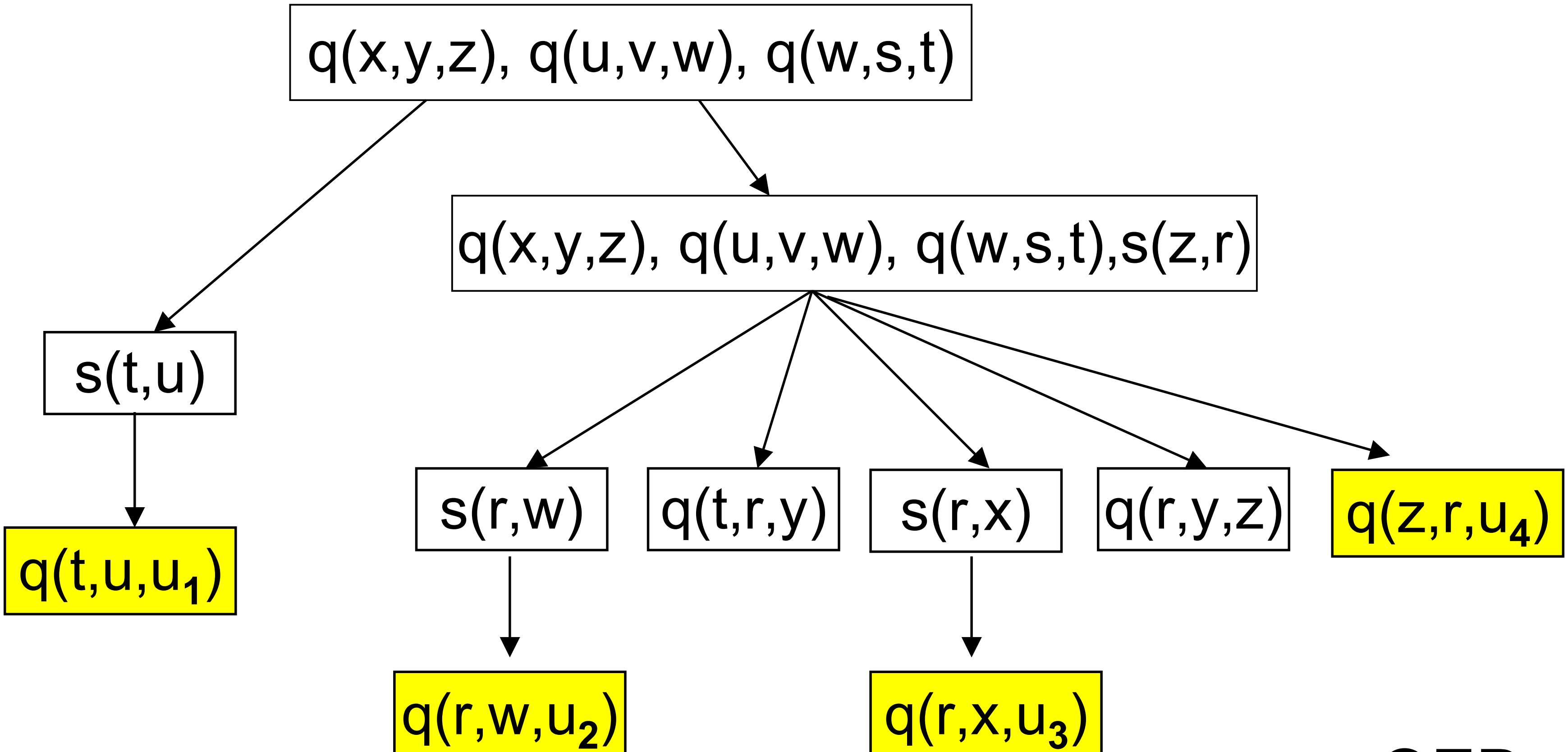


Proof Sketch (continued).

simple TGDs preserve hw

$s(x_1,x_2) \rightarrow \exists x_3\ q(x_1,x_2,x_3)$

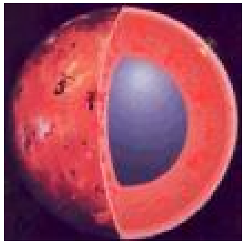
Variables from different blocks are never mixed!



Blocksize can increase, but hw remains bounded !

QED

Corollary: Core computation in data exchange is tractable if Σ_t consists of FDs and acyclic inclusion dependencies.



Tractability Result 2

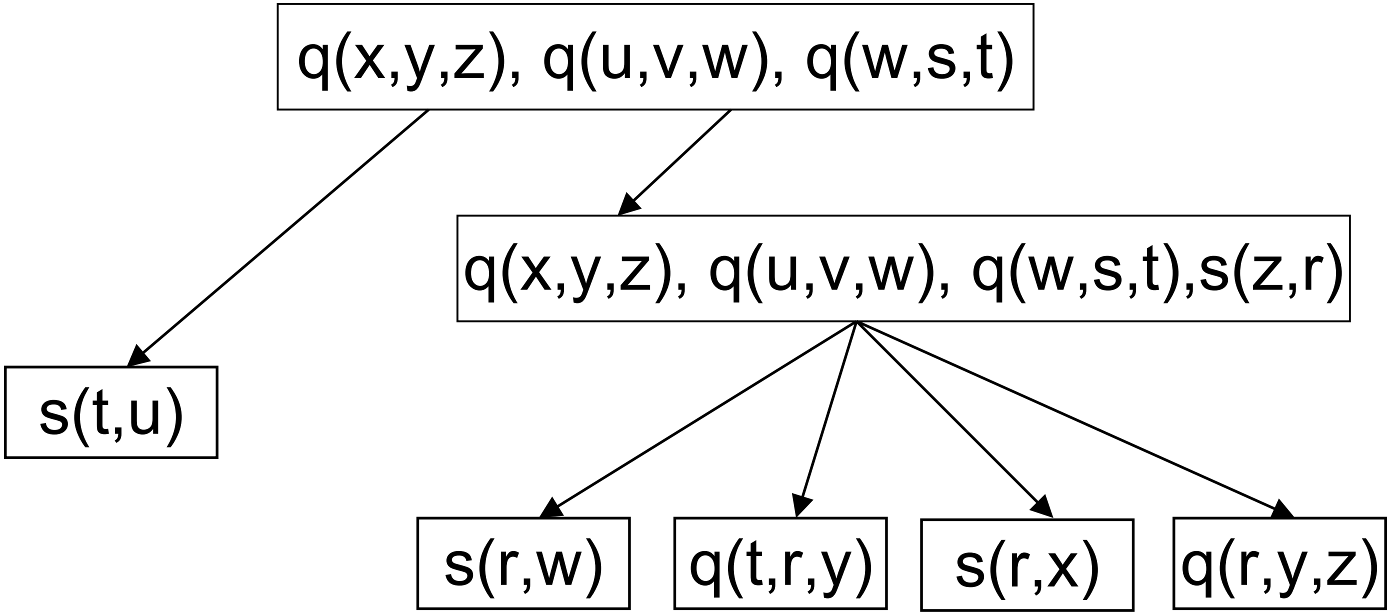
THEOREM: Data exchange with full TGDs + EGDs is tractable.

Difficulties

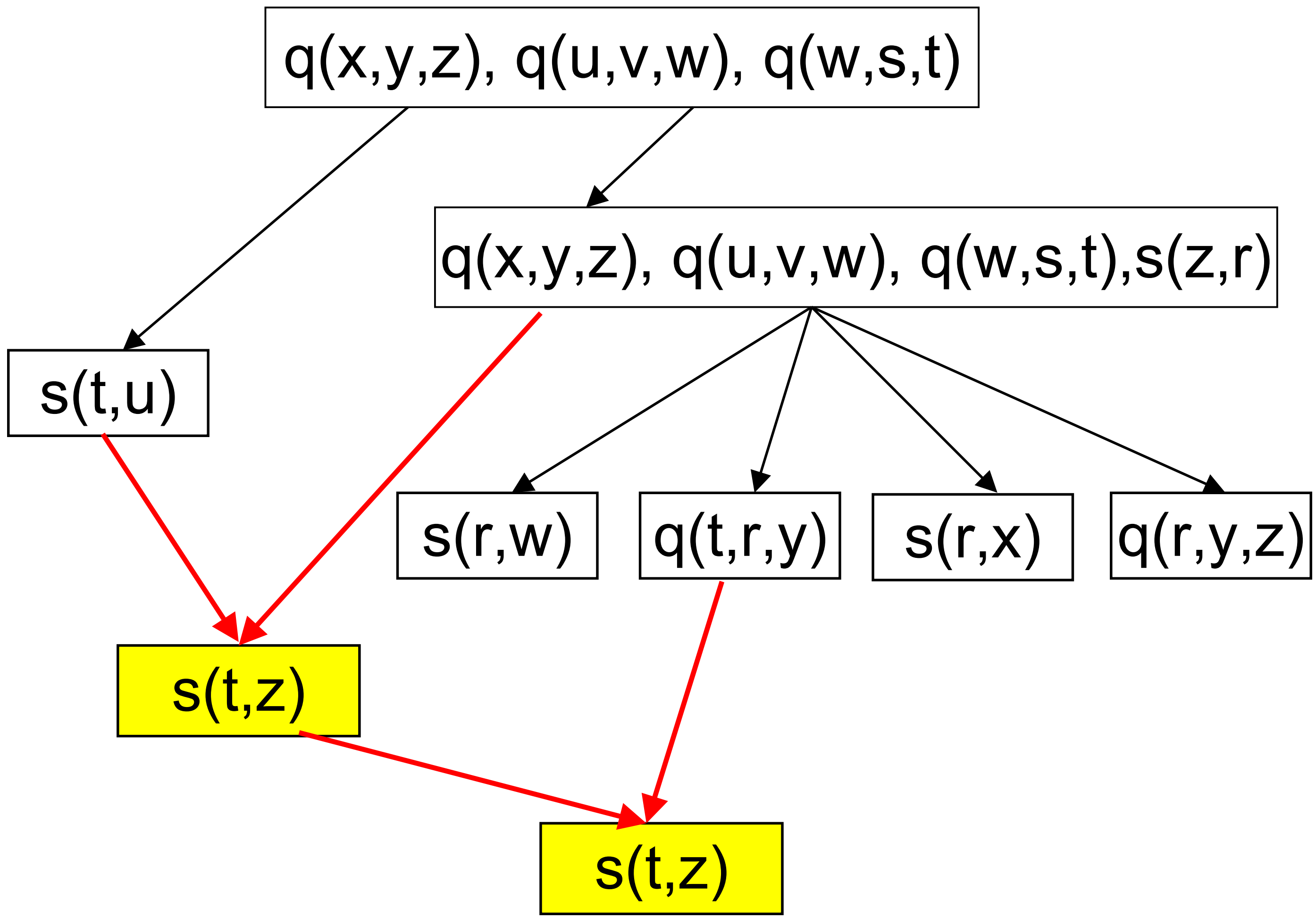
Blocks are merged when applying TGDs, thus no tree-structure possible !!!

➔Unbounded hypertee width.

Full TGD: $s(x_1,x_2)$ & $q(x_3,x_4,x_5) \rightarrow s(x_1,x_5)$



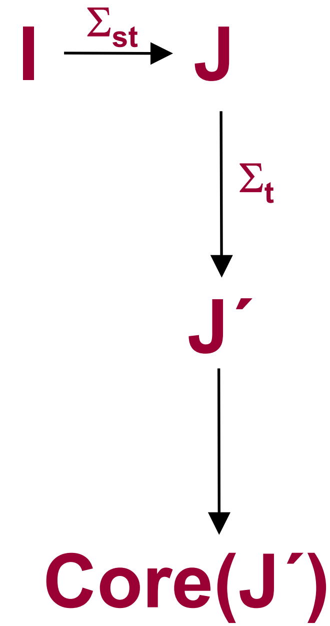
Full TGD: $s(x_1,x_2)$ & $q(x_3,x_4,x_5) \rightarrow s(x_1,x_5)$



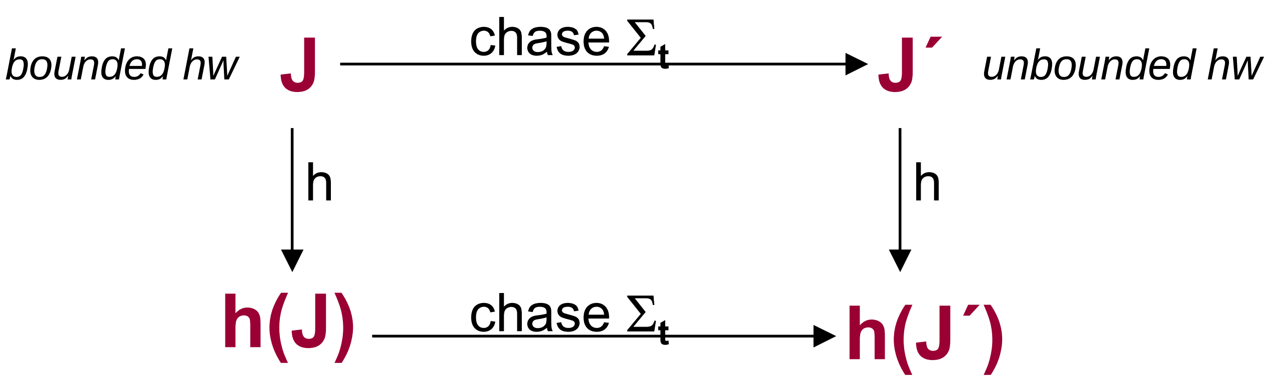
Solution:

Find useful homomorphism h: J→J'

such that h is a useful endomorphism J'→J'

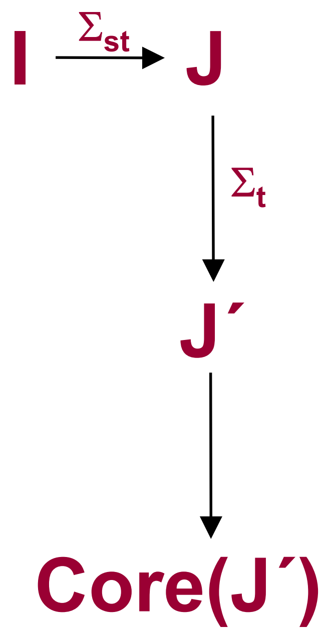


Possible, if the following diagram commutes:

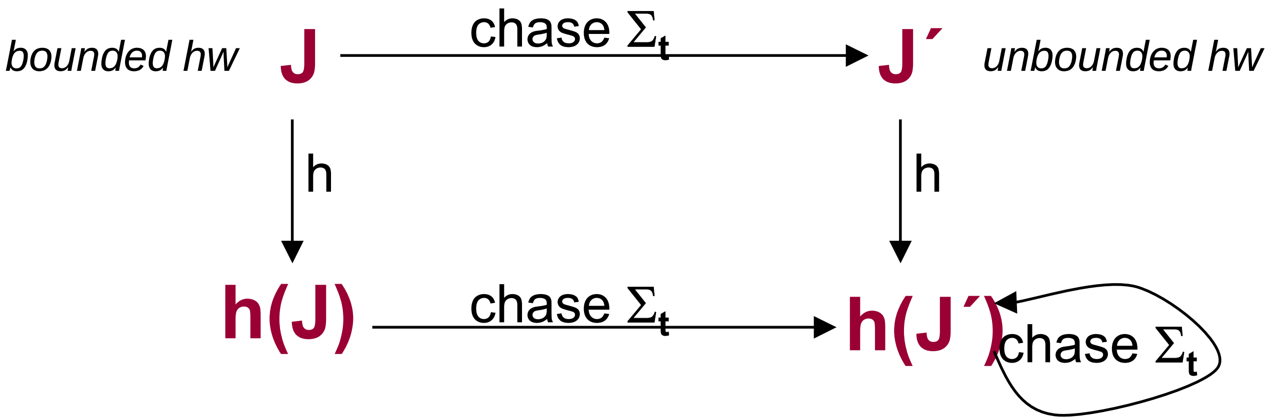


Solution:

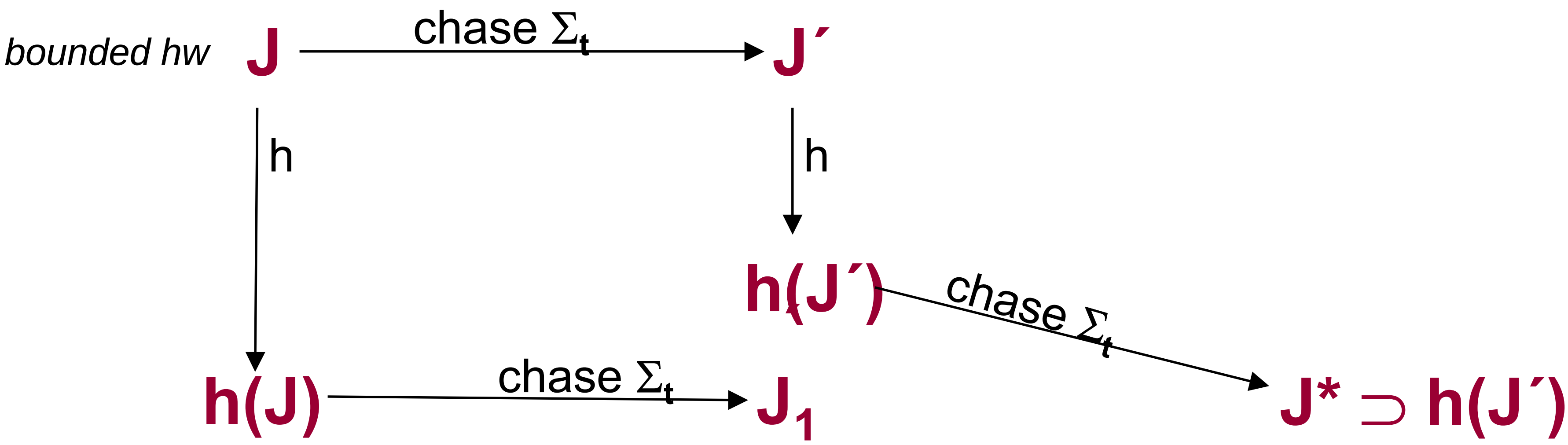
Find useful homomorphism h: J→J'
such that h is a useful endomorphism J'→J'



Possible, if the following diagram commutes:



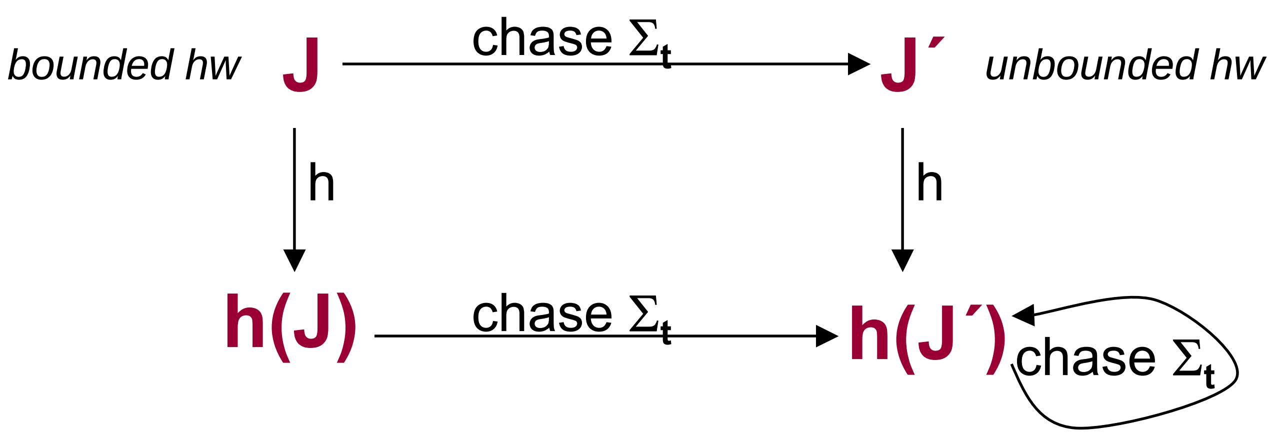
Unfortunately, in general, the diagram does not commute:



Definition: h idempotent if $h^2=h$, i.e., if $\forall x$: $h(h(x))=h(x)$

Lemma: If h is a useful endomorphism J→J', then h can be transformed in polynomial time into useful idempotent endomorphism J→J'.

Lemma: The diagram commutes whenever $h: J \rightarrow J'$ is an idempotent endomorphism.



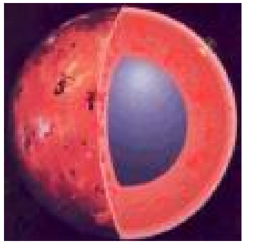
This allows us to design an algorithm that successively refines h via “local changes” as long as possible.
At the end, core(J) is computed.

To deal properly with EGDs, we simulate EGDs by full TGDs:

Assume the schema {p(.,.,.) s(.,.)}

p(x,y,z) → x=y

is simulated by the TGDs



p(x,y,z) & s(x,y) → s(x,x)
p(x,y,z) & s(x,y) → s(y,y)
p(x,y,z) & p(x,u,y) → p(x,u,x)
p(x,y,z) & p(x,u,y) → p(y,u,y)
etc. …..

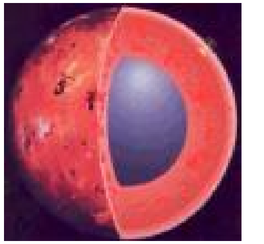
We can show that after computing the core these TGDs have the same effect as the EGD.

Limits of Tractability

Theorem: Data exchange with full TGDs and a Null predicate is NP-complete.

P(X,a) & q(X,Y) & Null(X) → r(X,Y)

Conclusions



- New general algorithm for computing cores based on hypertree decompositions

- A relevant class of target dependencies, for which data exchange is tractable:

  EGDs + weakly acyclic simple TGDs
  (encompassing FDs + w.a. INDs)

- A more involved tractable class: EGDs + full TGDs

Open: EGDs + weakly acyclic TGDs.