Deductive Databases

Semantic Optimization of Queries

2020/2021

Outline

- Semantic Query Optimization (SQO)
  - A quick refresher of basic notions
- Two-Phased Approach to SQO
  - Semantic compilation
  - Example scenarios
- Partial Subsumption
- Extended forms
- Semantically Constrained Axioms

Semantic Query Optimization

- Use semantic knowledge (e.g. integrity constraints) for transforming a query in a more efficient form than the original version
- The strategy chosen for optimization and its cost are important considerations
  - We will study a two-phased approach to semantic query optimization

A quick refresher of basic notions

- A literal is either an atomic formula or the negation of an atomic formula
- A clause is a disjunction of literals which has the form

$$S_1 \vee \ldots \vee S_m \vee \neg R_1 \vee \ldots \vee \neg R_n$$

  - where each S_i and R_j are atomic formulas
- Clauses can be written in an equivalent form using implication
  - $S_1 , \ldots, S_m \leftarrow R_1, \ldots, R_n$
  - All the variables in a clause are assumed to be universally quantified

A quick refresher of basic notions

- In logic programming, a program consists of clauses
  - Prolog restricts clauses to be Horn
  - They are adequate for specifying a large class of database applications
- A clause is Horn if $m \leq 1$ and disjunctive otherwise

$$S \leftarrow R_1, \ldots, R_n$$

  - where each S and R_j are atomic formulas
- A Horn clause is one whose head consists of at most a single atom

A quick refresher of basic notions

- A Horn clause is one whose head consists of at most a single atom
  - A goal clause has a null head;
  - The null clause has both a null body and a null head;
    - It represents a contradiction
  - A clause is definite if the head of the clause contains exactly one atom;
  - A unit clause is a definite clause with a null body;
  - A ground unit clause is a unit clause all of whose arguments are constants.

A quick refresher of basic notions

- We use operators to denote relations corresponding to widely used arithmetic operations
  - such as >, <, =, ≠, ≤, ≥
  - These relations are termed evaluable (or built-in) relations in contrast to (nonevaluable) relations defined as part of the database
- A clause is range-restricted if every variable in the head also appears in the body as arguments of nonevaluable relations
- A clause is called recursive if the same relation symbol appears both in the head and the body

A quick refresher of basic notions

- The notion of deductive database is based on the proof-theoretic approach to databases
- We limit our discussion to the basic components of a deductive databases
  - Facts
  - Deductive Rules (Axioms)
  - Integrity Constraints

Facts

- Database facts are expressed as ground unit clauses
  - The fact that a tuple $< a_1, \ldots, a_n >$ is a member of the relation R is expressed by the atomic literal

$$R(a_1, \ldots, a_n) \leftarrow$$

- Such facts comprise the extensional database, EDB, and such a relation R is called an extensional (stored) relation

Deductive Rules

- The intensional database, IDB, contains deductive rules
  - Relations defined by deductive rules are called intensional relations

Horn clause

Emp-Dept(ename,dname)
    ← Emp(eid,ename,did), Dept(did,dname)

  - This is a rule defining the (intensional) Emp-Dept relation in terms of the (extensional) Emp and Dept relations

- In general, a rule

$$S \leftarrow R_1, \ldots, R_n$$

- defines a relation S in terms of the relations $R_1, \ldots, R_n$

Integrity Constraints

- Integrity constraints contain rules expressing restrictions that the database must satisfy
  - A large class of integrity constraints are expressible as Horn clauses
- The following are examples on the extensional relation Supplier(Sno,Sname,Item)

Integrity Constraints

$y1=y2 \leftarrow Supplier(x, y1, z1), Supplier(x, y2, z2)$

$\leftarrow Supplier(x, y, \text{‘gun’}), Supplier(x, z, \text{‘butter’})$

Integrity Constraints: Examples

$y1=y2 \leftarrow Supplier(x, y1, z1), Supplier(x, y2, z2)$

- It represents the functional dependency of supplier name on supplier number

$\leftarrow Supplier(x, y, \text{'gun'}), Supplier(x, z, \text{'butter'})$

- It states that no supplier supplies both the object ‘gun’ and the object ‘butter’

Integrity Constraints

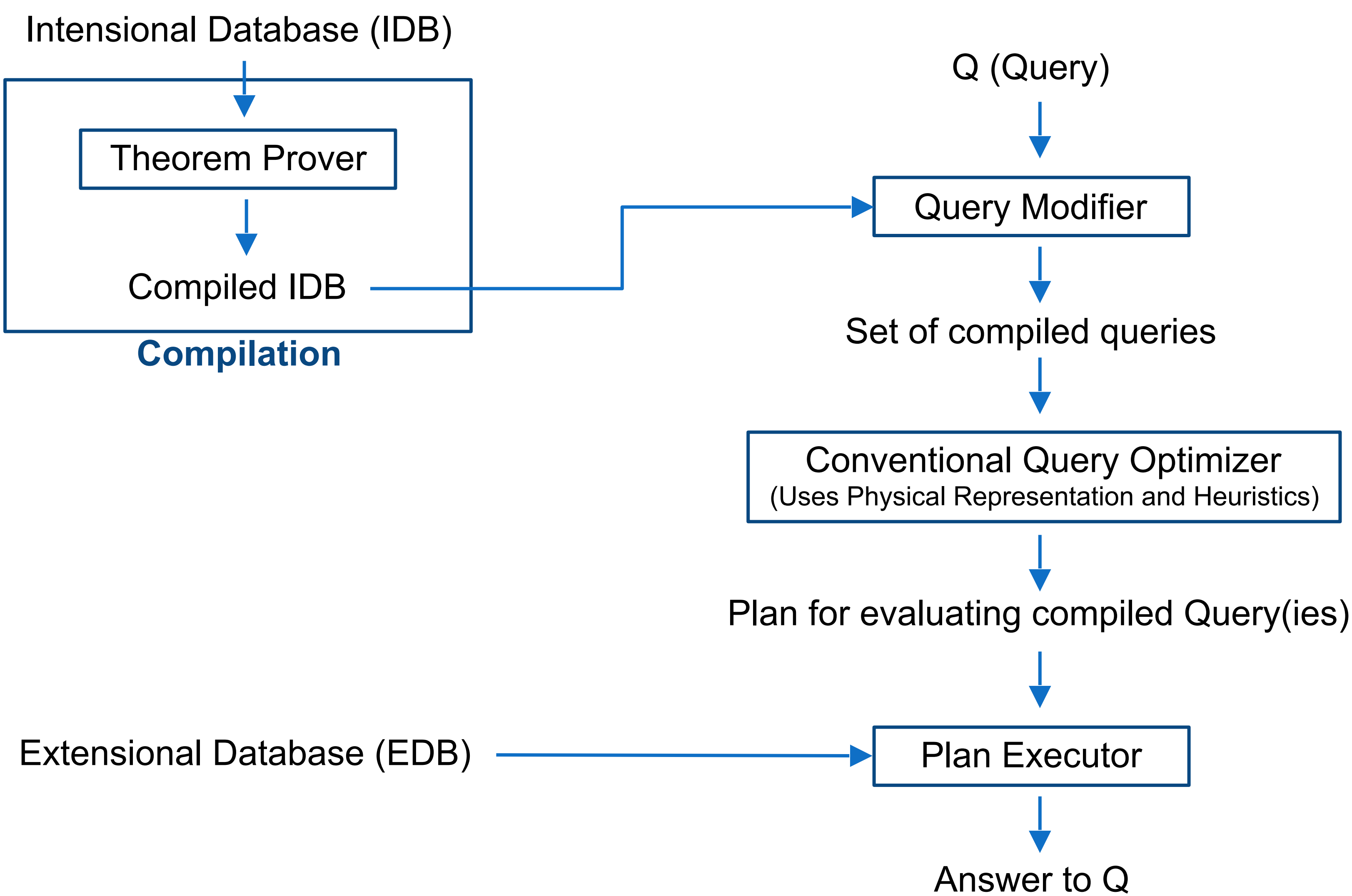
- Integrity constraints play an important role in checking update validity
- They are not needed in answering queries over definite deductive databases
- Semantic query optimization builds upon the premise that integrity constraints are useful for query evaluation
  - They can be used in transforming a query into a form that is more efficient to evaluate

Compiled Approach

- In (non-recursive) deductive databases, query evaluation can be partitioned into
  - a compilation phase followed by a modification and an evaluation phase

- With the compiled approach a query is evaluated by
  - first modifying the query using the compiled (intensional) database, and then
  - optimizing the result using conventional techniques

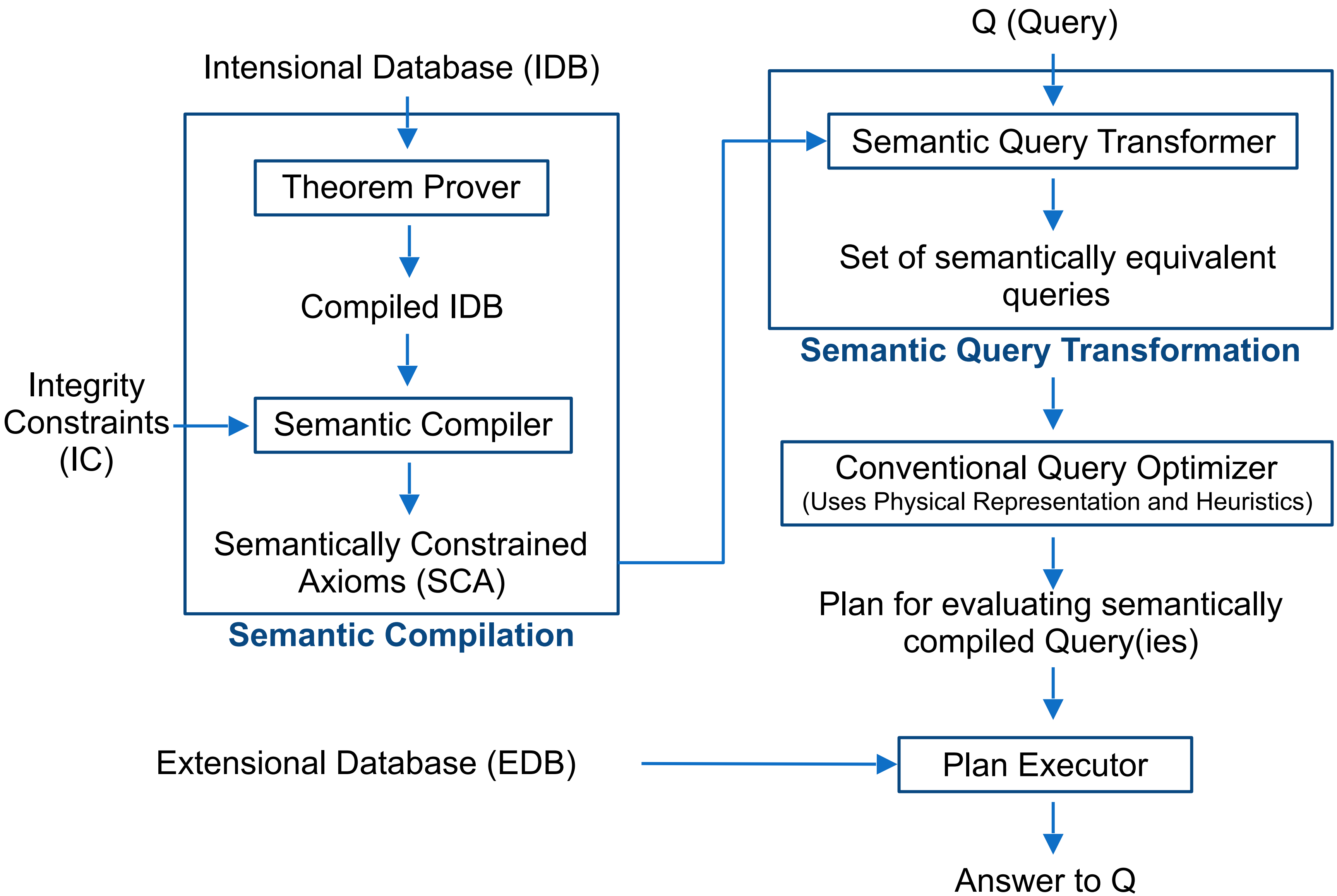
Compiled Approach



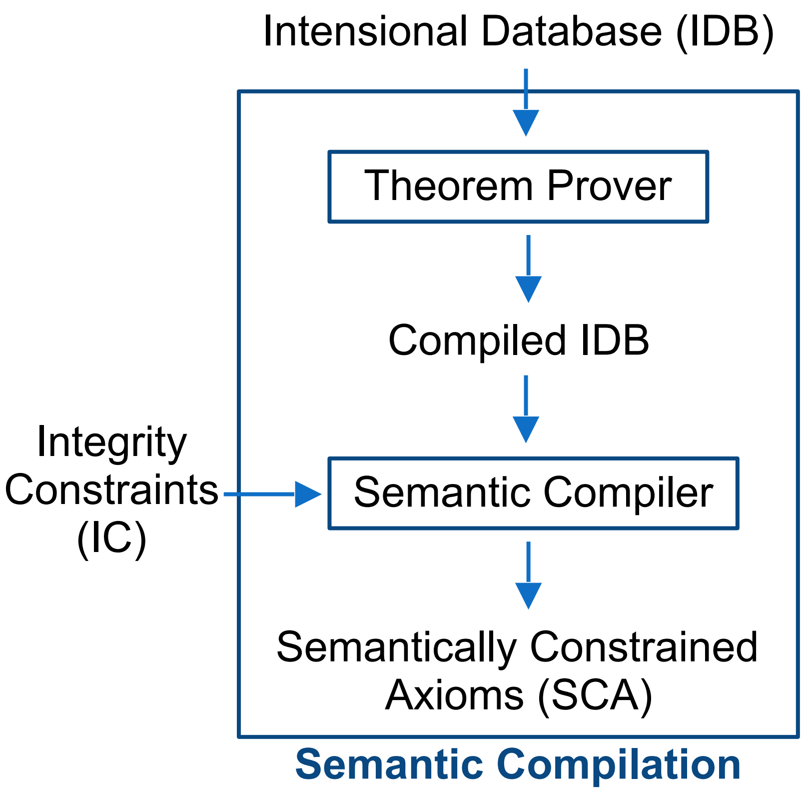
Two-Phased Approach to SQO

- The Two-Phased approach
  - Uses integrity constraints for optimizing a query over a deductive database
  - Generates several semantically equivalent queries as a result of the query modification stage, instead of a single query
- This entails changes to both
  - the compilation stage (now, semantic compilation phase)
  - the query modification stage (now, semantic transformation phase)

Two-Phased Approach to SQO

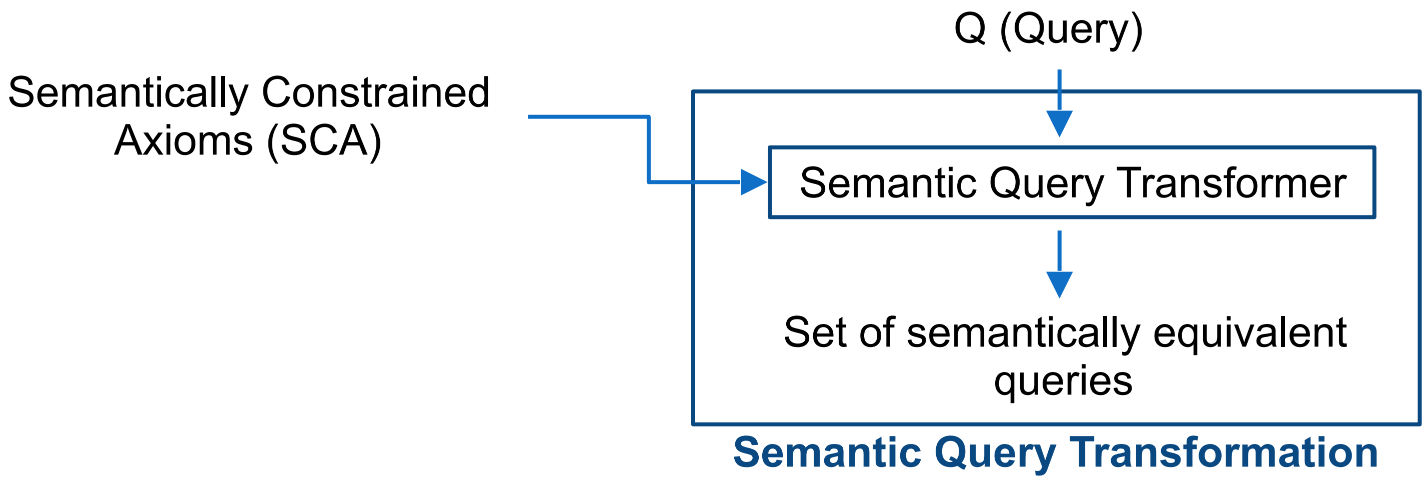


Two-Phased Approach to SQO



- During the semantic compilation phase, integrity constraint fragments, called residues, are computed and associated with deductive rules
- The result is a set of semantically constrained axioms (SCA) which are filtered and stored for later use

Two-Phased Approach to SQO



- When a query is given to the system, the semantic transformer uses these stored residues to generate semantically equivalent queries that *may be processed faster than the original query*

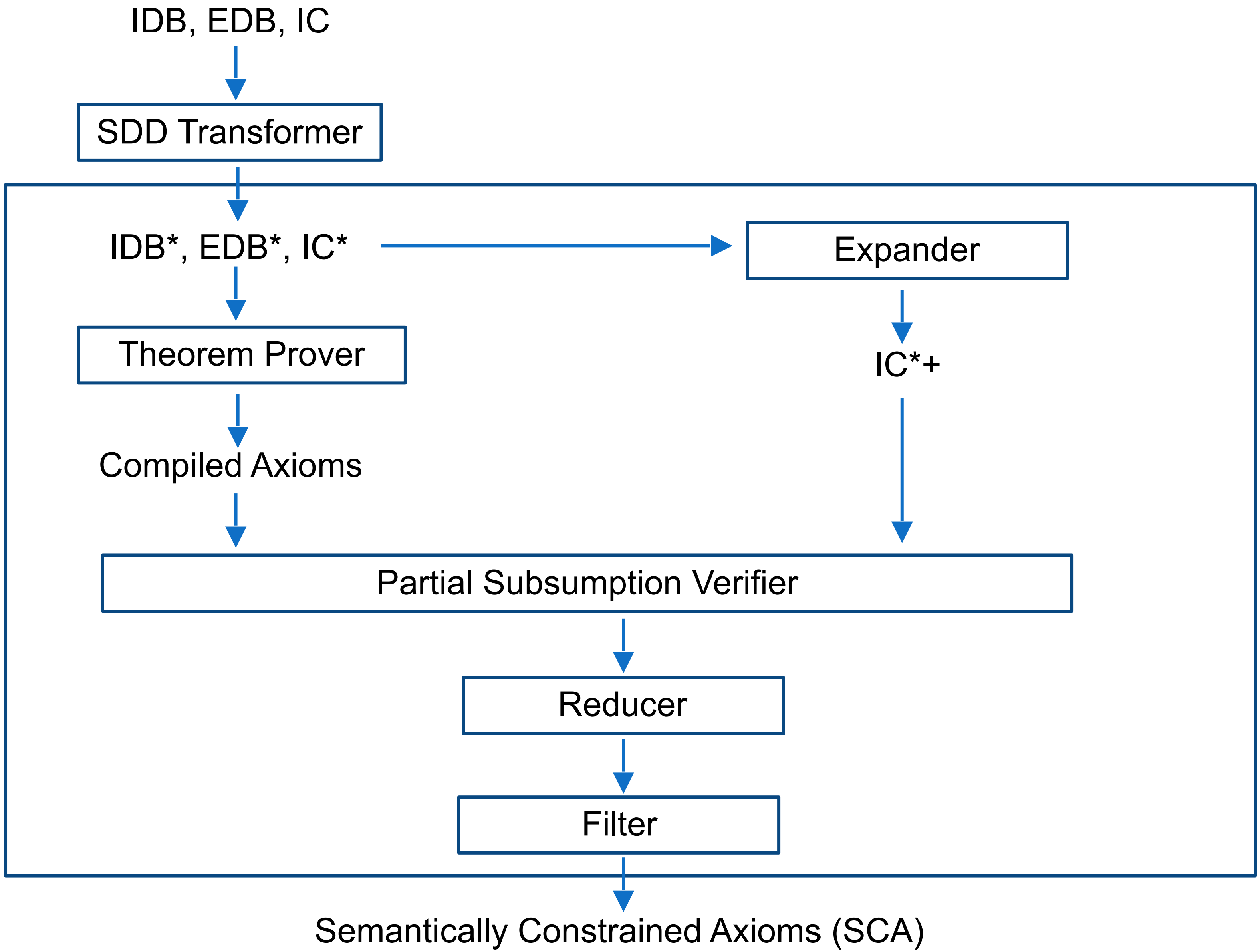
Types of Transformations

- Transformations to queries are performed using various heuristics
  - Literal elimination
    - hence a join, which is an expensive relational operation
  - Restriction introduction
    - may make it possible to use a range search, thereby reducing the cost of evaluation
  - Literal introduction
    - in some cases the introduction of a small instantiated relation for an additional join can reduce the amount of computation
  - Transformation that answers the query
  - Detection of unsatisfiable conditions

Semantic Compilation

- Semantic compilation can be described informally as the process of retaining relevant (or processed) fragments of integrity constraints
  - A step in which all useful information has been extracted from the integrity constraints
- Semantic compilation is performed only once for a given deductive (or relational) database
  - Is used over relatively stable database components
- The primary objective of semantic compilation is to associate integrity constraint fragments, called residues, with compiled axioms

Semantic Compilation Architecture



An example scenario

EDB: Ships(sname,owner,stype,draft,deadweight,capacity, registry)

Owners(oname,location,assets,business)

IC:

IC1: ← Owners(x1,‘iceland’,x3,x4)

IC2: x2 = ‘onassis’ ← Ships(x1,x2, ‘supertanker’,x4,x5,x6,x7)

Q:

Q1: ← Owners(x1*,‘iceland’,x3,x4*), (x3>1000000)

Q2: ← Ships(x1,x2,‘superthanker’,x4,x5,x6,x7*)

Q3: ← Owners(y1*,y2,y3,y4), Ships(x1,y1*,‘superthanker’,x4,x5,x6,x7)

An example scenario

EDB:

Ships(sname,owner,stype,draft,deadweight,capacity, registry)

Owners(oname,location,assets,business)

IC1: $\leftarrow$ Owners(x1,'iceland',x3,x4)

- «There are no owners with 'iceland' as their business location»

IC2: x2 = 'onassis' $\leftarrow$ Ships(x1,x2, 'supertanker',x4,x5,x6,x7)

- «All 'supertankers' are owned by 'onassis'»

- We show now how these IC may be useful in query processing
  - IC1 is useful only if Owners appears in the query, and
  - IC2 is useful only if Ships appears in the query

An example scenario

- Let now suppose the following query

Q1: $\leftarrow$ Owners(x1*,'iceland',x3,x4*), (x3>1000000)

  - It asks for the names and businesses of owners who are located in 'Iceland' and whose assets are greater than 1 million

- A relational database system may solve such a query by a table lookup of the Owners relation or by using an index on the Owners relation
- However, it follows from IC1 that Q1 cannot have any answers, so no search is needed at all

An example scenario

- Let now suppose the following query

Q2: $\leftarrow$ Ships(x1,x2,‘superthanker’,x4,x5,x6,x7*)

  - It asks for the registry information of all ‘supertankers’

- Assuming that the relation is not indexed on stype, a table lookup is necessary
- However, by using IC2 it is sufficient to look for x2 values of ‘onassis’ and the corresponding registry information
  - IC2 introduces a selection criterion for the owner attribute

An example scenario

- Let now suppose the following query

Q3: $\leftarrow$ Owners(y1*,y2,y3,y4), Ships(x1,y1*,'superthanker',x4,x5,x6,x7)

  - It asks for the owner names of all ships of the type 'supertanker'

- Using IC2 one can establish the value of y1* as 'onassis'.
  - However, it is necessary to confirm the presence of such a tuple in both the relations Ships and Owner
- An index is introduced on a join attribute, which also happens to be the output attribute in this example
  - In the presence of appropriate existential and inclusion integrity constraints, both the join and the lookup can be eliminated

Another example scenario

EDB:

Emp(ssno,salary,deptno,age)

Dept(deptno,manager,floor)

Sales(deptno,item,vol)

IDB:

Highsales(x1,x2,x3,y2,y3) ← Dept(x1,x2,x3),Sales(x1,y2,y3),y3>100000

HMgrProf(x2,y2,y4) ← Emp(x2,y2,y3,y4), Highsales(x1,x2,x3,x4,x5)

IC:

IC1: ← Dept(x,y,2)

IC2: (y>40000)← Emp(x,y,z,u),(u>50)

Q:

Q1: ← Highsales(x*,y,2,z*,u)

Another example scenario

- A standard method for dealing with such a database is to reduce all definitions of intensional relations to extensional relations
  - As long as the axioms (rules) in IDB are not recursive, the body of every axiom can be reduced to purely extensional relations using the compiled method

$$\begin{aligned}&HMgrProf(x1,x2,x4) \leftarrow \\ &\qquad Emp(x1,x2,x3,x4), Dept(y1,x1,y2), Sales(y1,z2,z3), z3>100000\end{aligned}$$

Another example scenario

- Let now suppose the following query

Q1: $\leftarrow$ Highsales(x*,y,2,z*,u)

  - It asks for the deptno and item values that are sold in large quantity (i.e., volume > 100000) by that department

  - A deductive database system may solve this query by transforming it using the rule for Highsales to

Q1’: $\leftarrow$ Dept(x*,y,2), Sales(x*,z*,u), (u>100000)

- It follows from IC1 and Q1’ that there are no answers

Subsumption

- A substitution (σ) is a finite set $\{t_1/v_1, \ldots, t_n/v_n\}$, where v_i 's are unique variables and t_i's are terms
  - Cσ is the clause obtained by replacing each occurrence of the variable v_i in C which is also in σ by the term t_i
- A clause C is a subclause of D if every literal in the clause C is also in D

A clause C subsumes a clause D if there is a substitution σ such that Cσ is a subclause of D

Subsumption

- Example 1

R(x,b)← P(x,y),Q(y,z,b)

R(a,b)← P(a,z),Q(z,z,b),S(a)

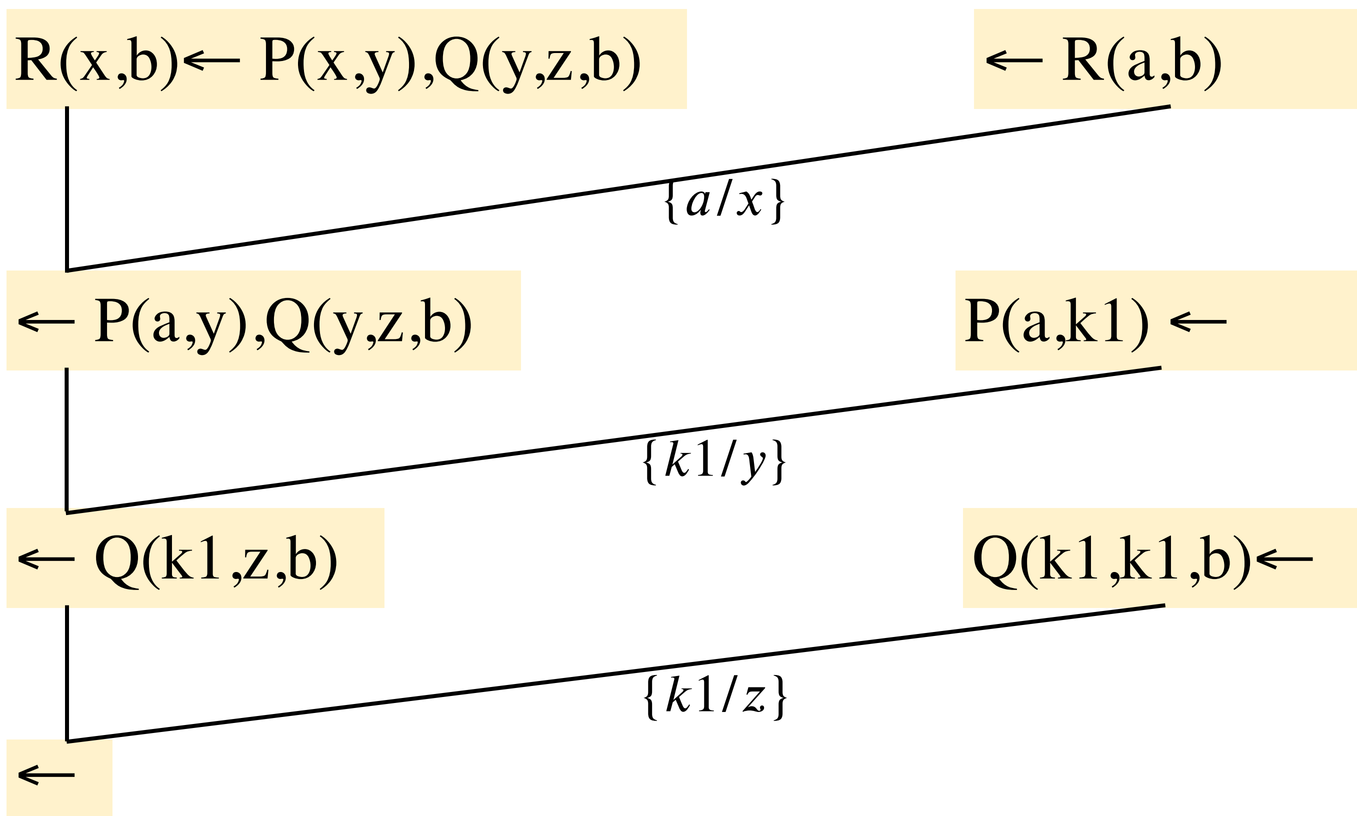
$\sigma = \{a/x,\ z/y\},$

- Example 2

IC1: ← Owners(x1,'iceland',x3,x4)

Q1: ← Owners(x1*,'iceland',x3,x4*), (x3>1000000)

Refusion Tree



R(x,b)← P(x,y),Q(y,z,b)

R(a,b)← P(a,k1),Q(k1,k1,b),S(a)

The Role of Partial Subsumption

- The semantic compilation phase occurs before any queries are posed
  - we must use the integrity constraints and the axioms for the relations to anticipate later modifications to queries
- The essence of partial subsumption is to apply the subsumption algorithm to an integrity constraint and the body of an axiom
  - a subclause of the integrity constraint might subsume the body of the axiom
  - Instead of the null clause, a fragment of the integrity constraint remains at the bottom of a refutation tree
    - Such a fragment is called a residue

The Role of Partial Subsumption

IC1: $\leftarrow$ Owners(x1,'iceland',x3,x4) (C)

Axiom: $\leftarrow$ Owners(y1,y2,y3,y4) (D)

Owners(x1,'iceland',x3,x4)

Owners(k1,k2,k3,k4) $\leftarrow$

?

- A variable is needed in place of the constant "iceland"
  - Consider C into the expanded form, denoted by C+, where C and C+ are logically equivalent

An IC ***partially subsumes*** *an axiom A if IC does not subsume the body of A, but a subclause of IC+ subsumes the body of A*

Expanded Forms

- An integrity constraint is expanded by substituting variables in place of constants systematically, in order to apply subsumption and resolve as many literals as possible

Steps of expansions

- The expanded clause C+ is obtained from the clause C by modifying the body of C as follows:
  1. An evaluable relation (predicate) that contains a constant and a variable is modified to two relations with the original relation containing two variables
     - (u > c) is replaced by (u > x1),(x1≥c)

Expanded Forms

Steps of expansions

- The expanded clause C+ is obtained from the clause C by modifying the body of C as follows:
  2. An extensional relation that contains a constant or a variable that has occurred previously (that is, to its left in the clause) is modified by changing the constant or the variable to a new variable and adding an equality consisting of the constant or the variable and the new variable introduced
     - IC1+: ← Owners(y1,x1,y3,y4), (x1='iceland')

Expanded Forms

IC1+: $\leftarrow$ Owners(y1,x1,y3,y4), (x1='iceland') (C)

Axiom: $\leftarrow$ Owners(y1,y2,y3,y4) (D)

$\leftarrow$ Owners(y1,x1,y3,y4), (x1='iceland')

Owners(k1,k2,k3,k4) $\leftarrow$

{k1/y1, k2/x1, k3/y3, k4/y4}

$\leftarrow$ k2='iceland'

- We can interpret this clause as "y2 cannot be iceland for Owners(yl,y2,y3,y4)"

Samantically Constrained Axiom

- Given an IC and an axiom A, apply the subsumption to IC+ and the body of A until no more resolutions are possible, and let B be the clause at the bottom of a refutation tree
- Then (B-)σ^{-1} that is, the clause obtained by the back substitution of the reduction of B is a residue of IC and A

← Owners(y1,x1,y3,y4), (x1=‘iceland’)

Owners(k1,k2,k3,k4) ←

σ={k1/y1, k2/x1, k3/y3, k4/y4}

← k2=‘iceland’

σ^{-1}

← y2=‘iceland’

Owners(y1,y2,y3,y4)← Owners(y1,y2,y3,y4), {← y2=‘iceland’}

Meaning of SCA

- A compiled intensional axiom H← P1, …, Pm (terms omitted) is transformed to its semantically constrained form H← P1, …, Pm {R1, …, Rm}, where the Ri are residues
- Let Ri have the form G← F1, …, Fk and define Ri' as not(F1, . . . , Fk, not(G))
  - In the case of a unit residue, Ri' is G
- For the empty residue, Ri' is fail
  - Then the use of the residue Ri entails the addition of Ri' to the definition of H.

An Example Scenario

EDB: Ships(z1,z2,z3,z4,z5,z6,z7)

Owners(y1,y2,y3,y4)

IC:

IC1: $\leftarrow$ Owners(y1,'iceland',y3,y4)

IC2: z2 = 'onassis' $\leftarrow$ Ships(z1,z2, 'supertanker',z4,z5,z6,z7)

IC+:

IC1+: $\leftarrow$ Owners(y1,x1,y3,y4), x1='iceland'

IC2+: z2 = 'onassis' $\leftarrow$ Ships(z1,z2,x1,z4,z5,z6,z7), x1='supertanker'

An Example Scenario

- For IC1+

← Owners(y1,x1,y3,y4), (x1=‘iceland’)

Owners(k1,k2,k3,k4) ←

{k1/y1, k2/x1, k3/y3, k4/y4}

← k2=‘iceland’

- For IC2+

z2 = ‘onassis’ ← Ships(z1,z2,x1,z4,z5,z6,z7),
x1=‘supertanker’

Ships(k1,k2,k3,k4,k5,k6,k7)←

{k1/z1, k2/z1, k3/z3, k4/z4, k5/z5, k6/z6, k7/z7}

k2=‘onassis’ ← k3=‘superthanker’

An Example Scenario

- The semantically constrained axioms (SCA) are:

SCA1: Ships(z1,z2,z3,z4,z5,z6,z7) ←
Ships(z1,z2,z3,z4,z5,z6,z7){z2='onassis' ← z3='superthanker'}

SCA2: Owners(y1,y2,y3,y4) ← Owners(y1,y2,y3,y4) {← y2='iceland'}

- The meanings of these SCA are as follows:

for SCA1:

Ships(z1,z2,z3,z4,z5,z6,z7) ←
Ships(z1,z2,z3,z4,z5,z6,z7), not(z3='superthanker', not(z2='onassis'))

for SCA2:

Owners(y1,y2,y3,y4) ← Owners(y1,y2,y3,y4) , not(y2='iceland')