Preference Relations

Information Systems M

Prof. Paolo Ciaccia

http://www-db.deis.unibo.it/courses/SI-M/

Beyond skylines

- Skyline queries represent only an, indeed remarkable, example of preference queries for which no scoring function is necessary
- The study of what are also known as qualitative (in opposition to "quantitative") preferences has its roots in the field of economy, in particular decision theory (where scores are called "utilities")
  - For more information and references, see [Fis99] on the web site
  - Research fields in which preferences play a relevant role are: Artificial Intelligence, Human-Computer Interaction, Decision-Support Systems, Autonomous Systems, Multi-Agent Systems, etc.
- The study of preferences has lead to several models, whose detailed study is however beyond the purpose of this course
- In the following we focus our attention on the so-called "binary relation" model, in which preferences are a set of pairs of tuples
- We start with some illustrative examples, then consider basic properties of the model, and finally describe two languages for preference specification

The voters' paradox

- Consider 3 friends (Ann, Joe and Tom) who rank, each one according to his/her own preferences, 3 movies: M1,M2, and M3
- In order to reach some consensus, they decide to integrate their preferences using the following "majority rule":

we collectively prefer Mi over Mj
if at least 2 of us have ranked Mi higher than Mj

Ann	Joe	Tom
M1	M3	M2
M2	M1	M3
M3	M2	M1

→ M1 is preferable to M2

→ M2 is preferable to M3

→ M3 is preferable to M1

No scoring function can be defined!

Irrational Behavior

(this example can be found in [Fis99])

- Consider the lottery **(a,p)**, which pays € a with probability p and nothing otherwise

Given two lotteries, which one will you choose to play?

- Many people(*) exhibit the following **cyclic** pattern of preferences:
  - **(€500, 7/24)** preferable to (€475, 8/24)
  - (€475, 8/24) preferable to (€450, 9/24)
  - (€450, 9/24) preferable to (€425, 10/24)
  - (€425, 10/24) preferable to (€400, 11/24)
  - (€400, 11/24) preferable to **(€500, 7/24)**

(*) A. Tversky. *Intransitivity of preferences*. Psychological Review 76 (1969), pp. 31-48

A non-paradoxical case

- Consider the following table:
  and the preference:

I prefer cinema C to C' iff they show the same movie and C costs less than C'

ID	Movie	Cinema	Price
C1	2001 A Space Odissey	Admiral	10
C2	2001 A Space Odissey	Astra1	12
C3	Wide Eyes Shut	Astra2	9
C4	Wide Eyes Shut	Odeon1	10
C5	Shining	Odeon2	12

- We have that C1 is preferred to C2 and C3 to C4; no other preferences can be derived
- Thus, a hypothetical scoring function S should assign an equal score to, say, C3 and C1, and to C3 and C2
  - Since there is no preference between C3 and the first two tuples
- This is impossible: S(C1) = S(C2) = S(C3) contradicts S(C1) > S(C2)!

Qualitative preferences

- Scoring functions are only a "quantitative" mean to define preferences
- A much more general (thus, powerful) approach is to consider so-called qualitative preferences

With **qualitative preferences** we just require that, given two tuples t1 and t2, there exists **some criterion** to determine whether t1 is preferred to t2 or not

- Since, when a scoring function is available, we prefer t1 to t2 iff S(t1) > S(t2), this shows that qualitative preferences are indeed a generalization of quantitative ones

A 1st game with qualitative preferences…

- This evening I would like to go out for dinner
- It's a special occasion, thus I'm willing to spend even up to 100 €, provided we go to a nice place (good atmosphere, good service and candle-lights), otherwise, say, 50 € would be the ideal target budget
- However, she really likes fish (which is quite expensive)
- As to the location, it would be better not to go downtown (too crowdy), she would love a place over the hills
- If the road is not too bad, I could also consider travelling for 1 hour, otherwise it would be preferable to travel for no more than ½ hour, say, so that coming back would be easier
- Formal dressing should not be required
- …
- Ok, let's start browsing the Yellow Pages…

A 2nd game with qualitative preferences…

- I would like to buy a used car
- I definitely do not like SUV's and would like to spend about 8,000 €
- Less important to me is the mileage
- Given this, it would be nice if the color is red and if the nominal fuel consumption is no more than 7 litres/100 km
- …

Preferences relations

- Consider a relation R(A1,A2,…,Am), and let
  Dom(R) = Dom(A1)xDom(A2)x…xDom(Am)
  be the domain of values of R (Dom(Ai) is the domain of Ai)
- A preference relation $\succ$ over R is a subset of Dom(R) x Dom(R), that is, a set of pairs of tuples in dom(R)
- If (t1,t2) $\in \succ$, we write t1 $\succ$ t2 and the meaning is that t1 is preferred to t2 (also: t1 dominates t2, t1 is better than t2)

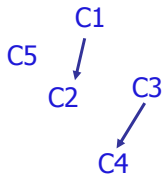
$\succ$ = {(C1,C2),(C3,C4)}

ID	Movie	Cinema	Price
C1	2001 A Space Odissey	Admiral	10
C2	2001 A Space Odissey	Astra1	12
C3	Wide Eyes Shut	Astra2	9
C4	Wide Eyes Shut	Odeon1	10
C5	Shining	Odeon2	12

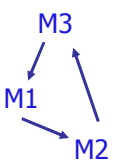
Preference graph

- A convenent way to represent a preference relation $\succ$ is through its directed graph **$G_{\succ}(V,E)$**, in which:
  - V = set of tuples and
  - E = {(t1,t2): t1 $\succ$ t2 }

ID	Movie	Cinema	Price
C1	2001 A Space Odissey	Admiral	10
C2	2001 A Space Odissey	Astra1	12
C3	Wide Eyes Shut	Astra2	9
C4	Wide Eyes Shut	Odeon1	10
C5	Shining	Odeon2	12



Ann	Joe	Tom
M1	M3	M2
M2	M1	M3
M3	M2	M1



Properties of a preference relation

- As any binary relation, a preference relation $\succ$ can be characterized in terms of some basic properties, thus $\succ$ can be:

Irreflexive iff:	$\forall t$:	$not(t \succ t) \equiv t \not\succ t$
Transitive iff:	$\forall t1,t2,t3$:	$(t1 \succ t2, t2 \succ t3) \Rightarrow t1 \succ t3$
Asymmetric iff:	$\forall t1,t2$:	$t1 \succ t2 \Rightarrow t2 \not\succ t1$
Negatively transitive iff:	$\forall t1,t2,t3$:	$(t1 \not\succ t2, t2 \not\succ t3) \Rightarrow t1 \not\succ t3$
Connected iff:	$\forall t1,t2$:	$t1 \succ t2 \vee t2 \succ t1 \vee t1 = t2$

- Not all these properties are independent of each other
- In particular:
  - $A \Rightarrow I$ If $t \succ t$, then A fails by taking $t1 = t2$
  - $I,T \Rightarrow A$ If $t1 \succ t2$ and $t2 \succ t1$, then either I or T fail:
    If T holds then I fails: $(t1 \succ t2, t2 \succ t1) \Rightarrow t1 \succ t1$
    If I holds then T fails: $(t1 \succ t2, t2 \succ t1) \not\Rightarrow t1 \succ t1$

Basic order types

- Of practical relevance are those combinations of properties that allow the definition of a kind of order among tuples
- The three relevant cases are:

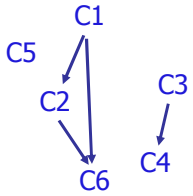
Strict partial order (I,A):
- A preference relation is a strict partial order (spo) iff it is transitive and irreflexive (thus, also asymmetric)

Weak order (I,A,N):
- A preference relation is a weak order (wo) iff it is a negatively transitive spo

Total order (I,A,C):
- A preference relation is a total order (to) iff it is a connected spo

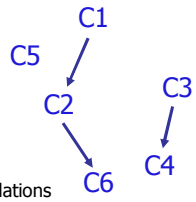
Spo: the preference graph

- If $\succ$ is an spo, then $G_{\succ}$ is acyclic

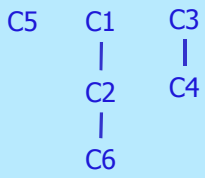


ID	Movie	Cinema	Price
C1	2001 A Space Odissey	Admiral	10
C2	2001 A Space Odissey	Astra1	12
C3	Wide Eyes Shut	Astra2	9
C4	Wide Eyes Shut	Odeon1	10
C5	Shining	Odeon2	12
C6	2001 A Space Odissey	Lumiere	13

- Since $\succ$ is transitive, it can be also drawn in a "transitively-reduced" form, by omitting all the edges that can be obtained by applying the transitivity rule

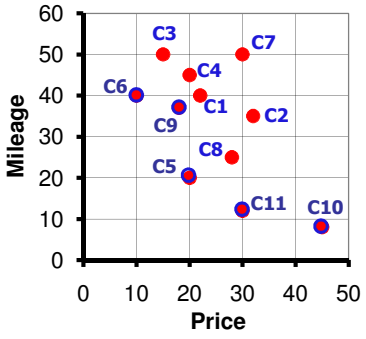


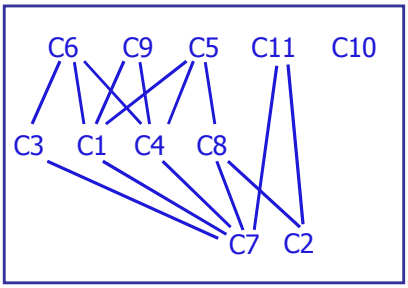
Hasse diagram:
Assuming that "higher" means "better", we can also draw the graph by omitting edge directions



Skyline dominance

- The skyline dominance rule is an spo (since it is both I and T)





BMO queries and the Best operator

- With preference relations the relevance/goodness of an object does not depend only on the object itself (as with scoring functions), rather it also depends on **all** other objects in the DB

S ≡ Absolute goodness ⇒ $\succ$ ≡ Relative goodness

- As a first step, and also generalizing skyline queries, we have the following definition [Cho02,Kie02,TC02]:

BMO queries and the Best operator:
Given a relation R and a preference relation $\succ$ over R, a Best-Matches-Only (BMO) query returns all the undominated tuples in R. This can be expressed through the Best (β) operator:

$$\beta_{\succ}(R) = \{t \in R \mid \forall t' \in R: t' \not\succ t\}$$

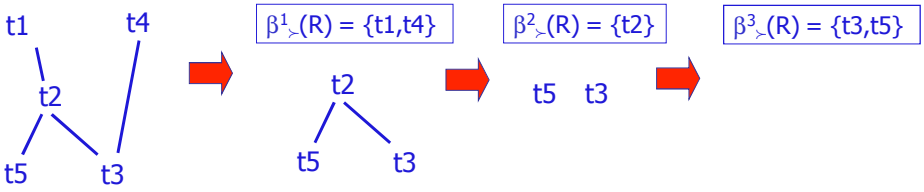
- Best is called "winnow" in [Cho02] and "preference selection" in [Kie02]

Ranking with Best

- Ranking of tuples can be easily obtained by iterating the Best operator
- Define:

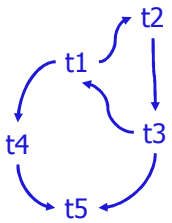
$$\beta^1_{\succ}(R) = \beta_{\succ}(R)$$
$$\beta^2_{\succ}(R) = \beta_{\succ}(R - \beta^1_{\succ}(R))$$
$$\beta^3_{\succ}(R) = \beta_{\succ}(R - \beta^1_{\succ}(R) - \beta^2_{\succ}(R))$$
$$\ldots$$

- Thus, $\beta^1_{\succ}(R)$ are the "top" tuples, $\beta^2_{\succ}(R)$ the "2nd" choices, and so on



Basic properties of the Best operator

- If $\succ$ is an spo then:
  1) $\beta_{\succ}(R)$ is always non-empty if R is non-empty (best tuples always exist)
  2) For each tuple $t \in R$ there is a level i such that $t \in \beta^{i}_{\succ}(R)$
     - This equals the lenght (no. of nodes) of the longest path leading from a tuple $\in \beta_{\succ}(R)$ to t
- If $\succ$ is not an spo, then it is possible to have $\beta_{\succ}(R) = \emptyset$, i.e. no undominated tuple exists



transitive and reflexive

- In this case a possible solution is to take all tuples in the "top cycles"
  - E.g., t1, t2, and t3 are "equally good", and all better than t4 and t5

Weak orders

A preference relation is a weak order (wo) iff it is a negatively transitive spo

$\forall t1,t2,t3: \ (t1 \not\succ t2, t2 \not\succ t3) \Rightarrow t1 \not\succ t3$

$\forall t1,t2,t3: \ t1 \succ t3 \Rightarrow (t1 \succ t2) \vee (t2 \succ t3)$

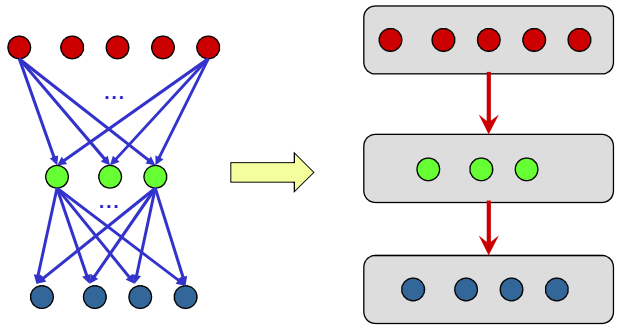
- The possible patterns (up to permutation of tuples) are:

- Thus, the forbidden configuration is:

i.e., if two tuples are ordered, then any other tuple must be ordered with respect to at least one of the two

Wo: the preference graph

- If $\succ$ is a wo, then $G_{\succ}$ is "layered":
  - All tuples within a same layer do not dominate each other
  - A tuple in a "higher" layer dominates all tuples in a "lower" layer
- If t1 $\succ$ t3, and t2 is in the same layer of t3, then it has to be: t1 → t2, t3



The indifference relation

- When we have both t1 $\not\succ$ t2 and t2 $\not\succ$ t1, we say that t1 and t2 are **indifferent**, written t1 $\sim$ t2
  - E.g., in the movies example we have C1 $\sim$ C3, C2 $\sim$ C3, etc..
- Since $\sim$ is a relation (called indifference relation), it can be characterized in terms of the properties it has
- By definition, $\sim$ is always Symmetric ($\forall$t1,t2: t1 $\sim$ t2 $\Rightarrow$ t2 $\sim$ t1)
- If $\succ$ is an spo, then $\sim$ is also Reflexive ($\forall$t: t $\sim$ t)

Weak order:
- A preference relation $\succ$ is a weak order (wo) iff its associated indifference relation $\sim$ is transitive, that is, an equivalence relation (R,S,T)

$$\forall t1,t2,t3: \quad (t1 \not\succ t2, t2 \not\succ t3) \Rightarrow t1 \not\succ t3$$
$$\underline{\forall t1,t2,t3: \quad (t2 \not\succ t1, t3 \not\succ t2) \Rightarrow t3 \not\succ t1}$$
$$\forall t1,t2,t3: \quad (t1 \sim t2, t2 \sim t3) \Rightarrow t1 \sim t3$$

Total orders

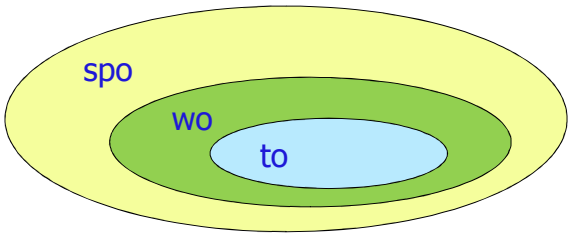
A preference relation is a total order (to) iff it is a connected spo

$$\forall t1,t2: \quad t1 \succ t2 \vee t2 \succ t1 \vee t1 = t2$$

- Every pair of tuples is ordered, thus $G_{\succ}$ is a "chain"

t1 ⟶ t2 ⟶ t3 ⟶ t4 ⟶ t5 ⟶ t6

- It is clear that a to is also a wo (in which no two tuples are indifferent)

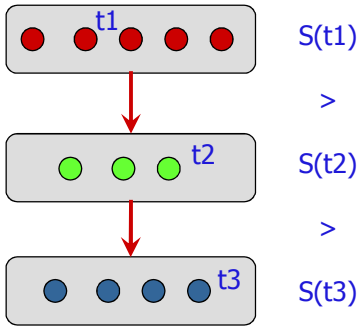


Weak orders and scoring functions

- A fundamental result asserts that :

Representability with a scoring function:
- A preference relation can be represented by a scoring function only if it is a weak order

- The intuition is that we can assign the same score to all tuples that are indifferent to each other



Non weak orders do not admit any S

- It is C1 ≻ C2 , C1 ∼ C3, C2 ∼ C3, thus ∼ is not transitive

ID	Movie	Cinema	Price
C1	2001 A Space Odissey	Admiral	10
C2	2001 A Space Odissey	Astra1	12
C3	Wide Eyes Shut	Astra2	9
C4	Wide Eyes Shut	Odeon1	10
C5	Shining	Odeon2	12
C6	2001 A Space Odissey	Lumiere	13

- Whatever S we choose, it will be:

S(C1) > S(C2)
S(C1) = S(C3)
S(C2) = S(C3)

that has no solution!

A wrong argumentation

C5 C1 C3
| |
C2 C4
|
C6

ID	Movie	Cinema	Price
C1	2001 A Space Odissey	Admiral	10
C2	2001 A Space Odissey	Astra1	12
C3	Wide Eyes Shut	Astra2	9
C4	Wide Eyes Shut	Odeon1	10
C5	Shining	Odeon2	12
C6	2001 A Space Odissey	Lumiere	13

- If we are interested in only the best (undominated) tuples, what's wrong in defining a scoring function S s.t. S(C1) = S(C3) = S(C5) > {S(C2), S(C4)} > S(C6)?

Answer: assume C1 is deleted:
Then S still yields:
S(C3) = S(C5) > {S(C2),S(C4)} > S(C6)
thus C2 is not one of the best objects!

ID	Movie	Cinema	Price
C2	2001 A Space Odissey	Astra1	12
C3	Wide Eyes Shut	Astra2	9
C4	Wide Eyes Shut	Odeon1	10
C5	Shining	Odeon2	12
C6	2001 A Space Odissey	Lumiere	13

On weak orders and scoring functions

- Not every weak order can be represented by a scoring function
- A sufficient condition is that Dom(R) be countable
- The classical counterexample (see also [Fis99]) goes as follows:

  Consider the order L on $[0,1]^2 \subset R^2$ (which is uncountable), defined by:

  $(x1,y1) \succ (x2,y2)$ if x1 > x2 or x1 = x2 and y1> y2

  Clearly, L is a weak order (it is also a total order).

  Assume there exists a scoring function S for L. This implies that:

  S(x1,1) > S(x1,0) > S(x2,1) > S(x2,0) whenever x1 > x2.

  Each interval (S(x,0), S(x ,1)) will then contain a (different) rational number, q(x).

  The function q maps from the real interval [0,1] to rational numbers, which leads to the contradiction that the countable set of rational numbers is uncountable.

- On the other hand, there are wo's on uncountable domains that can be represented by a scoring function (e.g. > on the real line)

Composition of preferences

- One of the most appealing aspects of qualitative preferences is that they provide a great flexibility when one comes to consider the issue of how different preference relations may be composed (combined)
- Understanding the rules governing preference composition is important from both a theoretical and practical point of view:
  - What if we add/drop some preferences?
  - What if we combine the preferences of different users/agents?
- By looking at the voters' paradox, it is evident that care has to be taken if one wants to preserve basic order properties
  - The preferences of each friend lead to a to, but their combination through the "majority rule" is not an spo
- For studying preference composition rules, we will use preference graphs as well as a specific formalism proposed in [Cho02] to specify preferences (i.e., a preference language)
  - We'll see another preference language later

A logical preference language [Cho02]

- In the logical language of J. Chomicki [Cho02], preferences are specified through a first-order formula P, such that:

$$t \succ_P t' \text{ iff } P(t,t')$$

  i.e., $t \succ t'$ iff the formula P is true when applied to t and t'

- The formula P can make use only of built-in predicates (it is a so-called "intrinsic preference formula", or IPF) , and as such can be evaluated by just looking at t and t' attribute values

Example:

- the formula P1: $(t.Movie = t'.Movie) \wedge (t.Price < t'Price)$ formalizes the preference for cheap cinemas we have used so far

A more complex example

- The preference "I prefer having white wine with fish and red wine with meat" on the DB

Menu

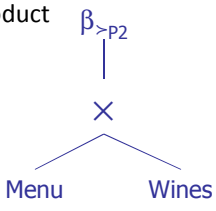
Dish	Dtype
Sea bass fillet	fish
Grilled prawns	fish
Steak tartare	meat
Hamburger	meat
T-bone steak	meat

Wines

Wine	Wtype
Chardonnay 2005	white
Pignoletto 2009	white
Chianti 2008	red
Merlot 2007	red
Rosato del Salento 2009	rosé

can be expressed, after performing the Cartesian product of the two relatons, as the formula:

P2: $(t.Dtype = \text{'fish'}) \wedge (t'.Dtype = \text{'fish'}) \wedge (t.Wtype = \text{'white'}) \wedge (t'.Wtype \neq \text{'white'})$
$\vee$ $(t.Dtype = \text{'meat'}) \wedge (t'.Dtype = \text{'meat'}) \wedge (t.Wtype = \text{'red'}) \wedge (t'.Wtype \neq \text{'red'})$



Checking properties of IPF's

- Order-theoretic properties of preferences resulting from IPF's can be verified as follows [Cho03]:
  1) Write down the negation of the property
  2) Try to solve the corresponding satisfiability (SAT) problem
  3) If no solution can be found, then the property holds

Examples:

irreflexivity: 1) $\exists$ t: t $\succ$ t, 2) solve P(t,t)

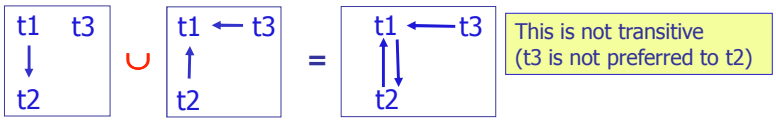
asymmetry: 1) $\exists$ t1,t2: t1 $\succ$ t2 $\wedge$ t2 $\succ$ t1, 2) solve P(t1,t2) $\wedge$ P(t2,t1)

transitivity: 1) $\exists$ t1,t2,t3: t1 $\succ$ t2 $\wedge$ t2 $\succ$ t3 $\wedge$ t1 $\not\succ$ t3,
2) solve P(t1,t2) $\wedge$ P(t2,t3) $\wedge$ $\neg$P(t1,t3)

Set-theoretic compositions: Union

- Consider 2 preference relations $\succ_{P1}$ and $\succ_{P2}$ over a same schema R
- Their union, $\succ_{P1} \cup \succ_{P2}$, corresponds to the IPF P1 $\vee$ P2

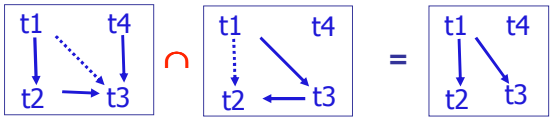
- If both $\succ_{P1}$ and $\succ_{P2}$ are spo's, this needs not to be the case for their union, since asymmetry and transitivity are not preserved. For instance:



- Spo's are preserved in the case of disjoint union, i.e., when no object is ordered by both $\succ_{P1}$ and $\succ_{P2}$
- Examples:
  - P1: (t.A = 'a' $\wedge$ t'.A = 'b'), P2: (t.A = 'c' $\wedge$ t'.A = 'd')
  - The "menu & wines" preference is a case of disjoint union
    - Preferences when eating fish are disjoint from those that apply when eating meat

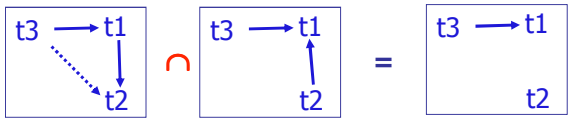
Set-theoretic compositions: Intersection

- $\succ_{P1} \cap \succ_{P2}$ corresponds to the IPF P1 ∧ P2
- Intersection preserves the spo properties. As an example:



Remind: the inputs are assumed to be spo's: dotted edges correspond to transitive preferences

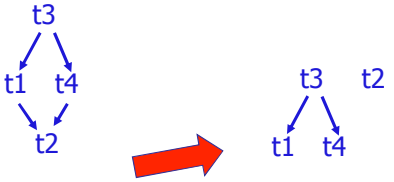
- With intersection the result is the set of preferences on which the two inputs agree, thus it cannot violate any of the properties of an spo
- If both preference relations are weak orders, their intersection needs not to be (in general, it is a strict partial order)



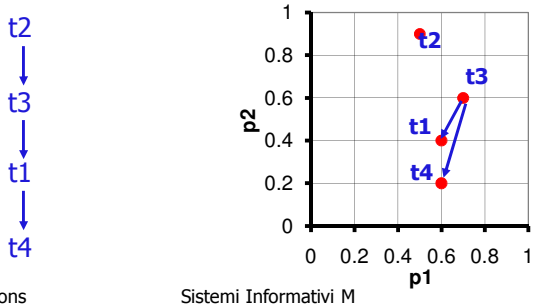
Intersection: from scoring functions to spo's

- We take the intersection of the following weak orders, each represented by a scoring function:

TID	p1
t3	0.7
t1	0.6
t4	0.6
t2	0.5

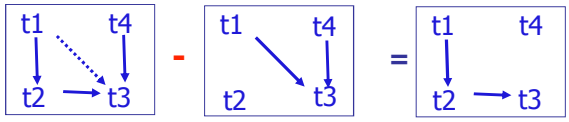


TID	p2
t2	0.9
t3	0.6
t1	0.4
t4	0.2

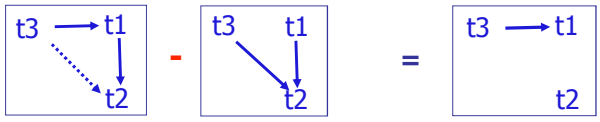


Set-theoretic compositions: Difference

- $\succ_{P1} - \succ_{P2}$ corresponds to the IPF $P1 \wedge \neg P2$
- Difference does not preserve transitivity:



- On the other hand, if the inputs are weak orders then the result is an spo:



Preference-specific rules

- Several preference-specific composition rules have been defined
- Among them, the two most important are:
- Prioritization rule: combines $\succ_{P1}$ and $\succ_{P2}$ by giving higher priority to, say, $\succ_{P1}$
- Pareto rule: combines $\succ_{P1}$ and $\succ_{P2}$ in a fair way
  - The idea is that t is preferred to t' iff it is never worse than t' and strictly better than t on at least one input preference
- For both rules there are 3 versions:
  - The "**composition**" version, that seems the most natural one, yet it does not preserve spo's
  - The "**accumulation**" one, which is too far restrictive
  - The "**accumulation with substitutable values**" one, which is a compromise among the first two

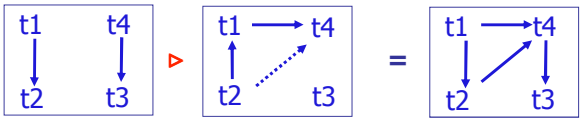
Prioritized composition (▷)

- Prioritized composition intuitively means:

  look first at $\succ_{P1}$, if no preference is given then look at $\succ_{P2}$

  $$t1 \succ_{P1} \triangleright \succ_{P2} t2 \equiv (t1 \succ_{P1} t2) \vee (t1 \sim_{P1} t2 \wedge t1 \succ_{P2} t2)$$

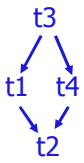
- If the inputs are weak orders, then the output is also a weak order (thus, it's ok for combining scoring functions)
- However, if the inputs are generic spo's, then their prioritized composition needs not to be, since transitivity is not preserved



Prioritization of scoring functions

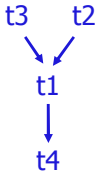
- We combine the following scoring functions, giving first priority to the first s.f. and then to the second one:

TID	p1
t3	0.7
t1	0.6
t4	0.6
t2	0.5



Priority is given to p1

TID	p2
t2	0.9
t3	0.9
t1	0.4
t4	0.2



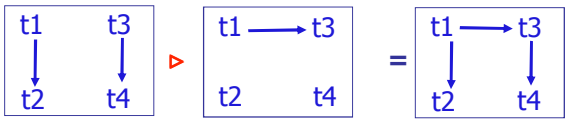
Priority is given to p2

Prioritization of strict partial orders

- Let P1: (t.Make = t'.Make) ∧ (t.Price < 10000) ∧ (t'.Price ≥ 10000)
  P2: (t.Price = t'.Price) ∧ (t.Make = 'BMW') ∧ (t'.Make ≠ 'BMW')
- The prioritized composition $\succ_{P1} \triangleright \succ_{P2}$ is not transitive:

- $\succ_{P1}$ = {(t1,t2),(t3,t4)}
- $\succ_{P2}$ = {(t1,t3)}
- $\succ_{P1} \triangleright \succ_{P2}$ = {(t1,t2),(t1,t3),(t3,t4)}

TID	Make	Price
t1	BMW	9000
t2	BMW	11000
t3	Audi	9000
t4	Audi	12000



Transitive closure

- As the previous example suggests, in some cases it would be desirable to be able to infer that t1 is better than t4
- To this end, one could compute the transitive closure (TC) of a preference relation:

The transitive closure of a preference relation $\succ_P$ is the relation $\succ_{P^*}$ defined as:

$$t1 \succ_{P^*} t2 \text{ iff } t1 \succ_P^n t2, \quad n > 0$$

where:

$$t1 \succ_P^1 t2 \equiv t1 \succ_P t2$$
$$t1 \succ_P^n t2 \equiv \exists t3: t1 \succ_P t3 \wedge t3 \succ_P^{n-1} t2$$

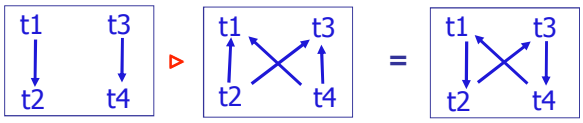
- The TC of an IPF is still an IPF (details omitted on how it can be derived in the general case)
- Note that taking the TC can lead to violate irreflexivity
  - This occurs if there are cycles in the preference graph of $\succ_P$

Prioritized composition: example of cycles

- Let P1: (t.Make = t'.Make) ∧ (t.Price < 10000) ∧ (t'.Price ≥ 10000)
  P2: (t.Price ≥ 10000) ∧ (t'.Price < 10000)

- $\succ_{P1}$ = {(t1,t2),(t3,t4)}
- $\succ_{P2}$ = {(t2,t1),(t2,t3),(t4,t1),(t4,t3)}
- $\succ_{P1} \triangleright \succ_{P2}$ = {(t1,t2),(t2,t3),(t3,t4),(t4,t1)}

TID	Make	Price
t1	BMW	9000
t2	BMW	12000
t3	Audi	9000
t4	Audi	12000



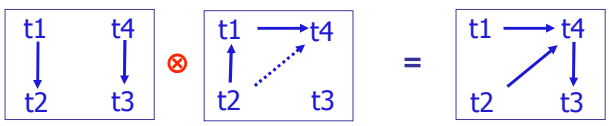
Pareto composition (⊗)

- With Pareto composition t is preferred to t' iff it is never worse than t' and strictly better than t' on at least one input preference

$$t1 \succ_{P1} \otimes \succ_{P2} t2 \equiv (t1 \succ_{P1} t2 \vee t1 \sim_{P1} t2) \wedge (t1 \succ_{P2} t2 \vee t1 \sim_{P2} t2) \wedge (t1 \succ_{P1} t2 \vee t1 \succ_{P2} t2)$$

- If the inputs are wo's, then the result is an spo
  - This is the **skyline dominance criterion**!

- If the inputs are generic spo's, then transitivity is not preserved

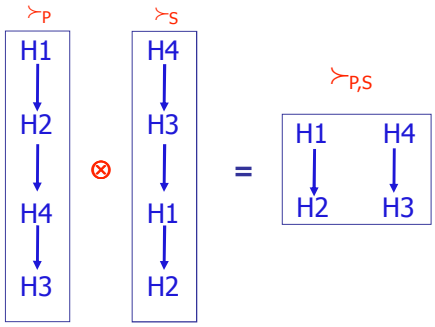


Pareto composition of weak orders

- Consider a set of hotels, each with a price (P), a number of stars (S), distance from the town center (D), and number of rooms (R)
- Let $\succ_P$ be the preference relation defined as: P : t.Price < t'.Price
- Let $\succ_S$ be defined as: S : t.Stars > t'.Stars

Hotel	Price	Stars	Distance	Rooms
H1	30 €	2	4 km	30
H2	35 €	1	2 km	20
H3	60 €	3	1 km	100
H4	40 €	4	6 km	50

Good and cheap hotels



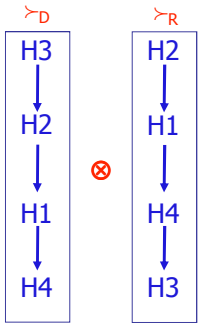
Pareto composition of strict partial orders

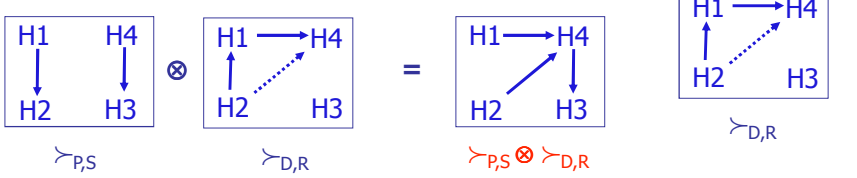
- Let $\succ_{P,S} = \succ_P \otimes \succ_S$
- Let $\succ_{D,R} = \succ_D \otimes \succ_R$, where:
- The preference relation $\succ_{P,S} \otimes \succ_{D,R}$ is not an spo!

Small and central hotels

D : t.Distance < t'.Distance
R : t.Rooms < t'.Rooms

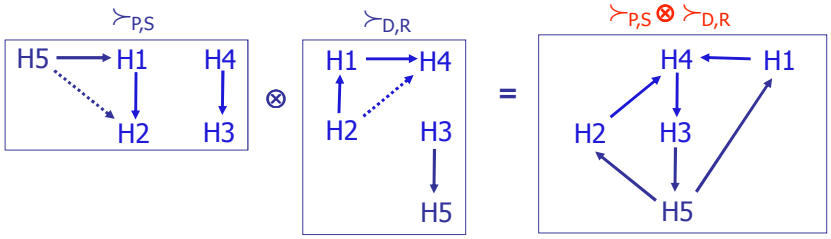
Hotel	Price	Stars	Distance	Rooms
H1	30 €	2	4 km	30
H2	35 €	1	2 km	20
H3	60 €	3	1 km	100
H4	40 €	4	6 km	50





… and cycles might arise!

Hotel	Price	Stars	Distance	Rooms
H1	30 €	2	4 km	30
H2	35 €	1	2 km	20
H3	60 €	3	1 km	100
H4	40 €	4	6 km	50
H5	**25 €**	**2**	**1.5 km**	**100**



Pareto and Prioritized accumulation

- This version of the two operators, respectively written as & (Pareto) and >> (Prioritization), is obtained by replacing indifference (∼) with equality (=) in the two definitions, and guarantees that spo properties are preserved:

$$t1 \succ_{P1} \gg \succ_{P2} t2 \equiv (t1 \succ_{P1} t2) \vee (t1 =_{P1} t2 \wedge t1 \succ_{P2} t2)$$

$$t1 \succ_{P1} \& \succ_{P2} t2 \equiv (t1 \succ_{P1} t2 \vee t1 =_{P1} t2) \wedge (t1 \succ_{P2} t2 \vee t1 =_{P2} t2) \wedge (t1 \succ_{P1} t2 \vee t1 \succ_{P2} t2)$$

- In both cases "=" means:
  "equality on the values of the attributes involved in P1 (or P2)"

Example: P1: (t.Price < 10000) ∧ (t'.Price ≥ 10000)
P2: (t.Make = 'BMW') ∧ (t'.Make ≠ 'BMW')

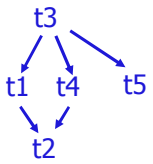
- $\succ_{P1}$ = {(t1,t2),(t3,t2)}
- $\succ_{P2}$ = {(t1,t3),(t2,t3)}
- $\succ_{P1} \gg \succ_{P2}$ = {(t1,t2),(t3,t2)}

TID	Make	Price
t1	BMW	8000
t2	BMW	11000
t3	Audi	9000

Substitutable values

- Equality is often a too restrictive condition
- On the other hand, using indifference (that is not an equivalence relation for generic spo's) in Pareto and Prioritized operators does not preserve spo's
- The idea of substitutable values/tuples is that both operators can be safely applied if indifference is replaced by an equivalence relation, which is called substitutabiliy and denoted $\approx$ [Kie05]:

Substitutable Values (SV):
Two tuples/values t1 and t2 are substitutable, $t1 \approx t2$, only if they:
- Are dominated by the same tuples
- Dominate the same tuples



t1 and t4 are substitutable
t1 and t5 are not

SV-equivalence: exampls

- Let P1: $(t.Make = t'.Make) \wedge (t.Price < 10000) \wedge (t'.Price \geq 10000)$
- All tuples t1 and t2 such that
  $(t1.Make = t2.Make) \wedge (t1.Price < 10000) \wedge (t2.Price < 10000)$
  can be considered substitutable
- The same is true if
  $(t1.Make = t2.Make) \wedge (t1.Price \geq 10000) \wedge (t2.Price \geq 10000)$
- On the other hand, if $t1.Make \neq t2.Make$ then t1 and t2 are **not** substitutable
  - Since they will dominate (be dominated by) different tuples

Pareto and Prioritized SV-accumulation

- This version of the two operators ($\&_{SV}$ and $>>_{SV}$) preserves spo's

$$t1 \succ_{P1} \gg_{SV} \succ_{P2} t2 \equiv (t1 \succ_{P1} t2) \vee (t1 \approx_{P1} t2 \wedge t1 \succ_{P2} t2)$$

$$t1 \succ_{P1} \&_{SV} \succ_{P2} t2 \equiv (t1 \succ_{P1} t2 \vee t1 \approx_{P1} t2) \wedge (t1 \succ_{P2} t2 \vee t1 \approx_{P2} t2) \wedge (t1 \succ_{P1} t2 \vee t1 \succ_{P2} t2)$$

- In both cases, t1 and t2 will still be equivalent only if they are so in both input preferences
- Intuitively: preferences are not sufficient to distinguish between t1 and t2

Example: P1: (t.Price < 10000) $\wedge$ (t'.Price $\geq$ 10000)
P2: (t.Make = 'BMW') $\wedge$ (t'.Make $\neq$ 'BMW')

- $\succ_{P1}$ = {(t1,t2),(t3,t2)}
- $\succ_{P2}$ = {(t1,t3),(t2,t3)}
- Now it is $\succ_{P1} \gg_{SV} \succ_{P2}$ = {(t1,t2),(t3,t2),**(t1,t3)**}

TID	Make	Price
t1	BMW	8000
t2	BMW	11000
t3	Audi	9000

An algebraic preference language

- Another major approach to DB-oriented preference specification is due to W. Kiessling [Kie02]
  - It consists of a set of "base constructors" + composition operators
- It is less powerful but more intuitive than IPF's, even because

All constructors and operators are designed so as to preserve spo properties

- In the following we present a slightly modified version of part of Kiessling algebra, which we call PA (Preference Algebra)

PA: Numerical base constructors

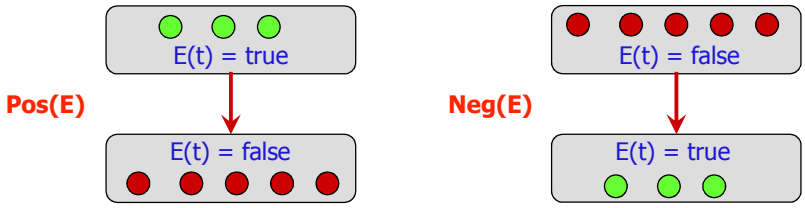
Constructor	Comment	Example
High(E)	higher values are better	High(Rating)
Low(E)	lower values are better	Low(10*Price + Rooms)
Around(E,v)	v is a "target value"	Around(Price, 40)
Between(E,[v1,v2])	[v1,v2] is a "target interval"	Between(Price,[30,40])

- E is a numerical expression
- Notice that High(E) = Low(-E) and Around(E,v) = Low(|E-v|)
- Between(E,[v1,v2]):
  - all values within the target interval are substitutable
  - t is better than t' iff E(t) is closer than E(t') to [v1,v2]
- In all cases a weak order is obtained

PA: Boolean base constructors

Constructor	Comment	Example
Pos(E)	values satisfying E are better	Pos(Price < 30)
Neg(E)	values not satisfying E are better	Neg(Cuisine='chinese' AND Price > 20)

- E is a Boolean expression
- Clearly, Neg(E) = Pos(not(E))
- In both cases a weak order with 2 levels is obtained



PA: "grouped" preferences

- For a set of attributes X, let **X↔** denote the "null preference" on X, which makes all values of X not ordered
- Then, the expression X↔ & P, where P is a preference expression, means:
  Apply P separately to each X value

Example: The expression Make↔ & Low(Price) corresponds to the IPF:
$(t.Make = t'.Make) \wedge (t.Price < t'.Price)$

- This can be generalized to **E↔**, where E is an expression, thus by "grouping" on different values of E

Example: The expression ⌊Price/10000⌋↔ puts in the same "group" all tuples whose price is in the range
[k*10000*Price,(k+1)*10000*Price), k > 0

PA: examples (1)

- Low(Price) $\&_{SV}$ High(Rating)
- (Pos(Cuisine='italian') $>>_{SV}$ Neg(Price>40 €)) $\&_{SV}$ Low(dist(Address,'Bologna'))
- ⌊Price/10⌋↔ $\&_{SV}$ Low(⌊Mileage/10⌋)
- (Pos(Style in {'SUV','coupe'}) $\&_{SV}$ Neg(Price>30)) $>>_{SV}$ Low(Price)
  $>>_{SV}$ (Pos(Color='red') $\&_{SV}$ Low(Mileage))
  - Let's work out the 4th expression, considering the following DB:

CarID	Make	Model	Style	Color	Price	Mileage
C1	Toyota	Corolla	sedan	Red	18	30
C2	BMW	325	coupe	Blue	35	20
C3	BMW	745	sedan	White	45	25
C4	Mercedes	CLK 5.0	coupe	Silver	40	35
C5	Porsche	Cayenne	SUV	Red	25	70
C6	Mercedes	CLK 5.0	coupe	Red	40	45
C7	Porsche	Cayenne	SUV	Black	30	60
C8	Nissan	350Z	coupe	Black	25	25
C9	VW	Passat GLS	sedan	Gray	15	35

PA: examples (2)

- We start by considering the two most important preferences:

  Pos(Style in {'SUV','coupe'}) $\&_{SV}$ Neg(Price>30)

- These define an spo with 4 classes of equivalent tuples:

CarID	Make	Model	Style	Color	Price	Mileage
C5	Porsche	Cayenne	SUV	Red	25	70
C6	Mercedes	CLK 5.0	coupe	Red	30	45
C7	Porsche	Cayenne	SUV	Black	30	60
C8	Nissan	350Z	coupe	Black	25	25

Style in {SUV,coupe} and Price ≤ 30

CarID	Make	Model	Style	Color	Price	Mileage
C1	Toyota	Corolla	sedan	Red	18	30
C9	VW	Passat GLS	sedan	Gray	15	35

Style not in {SUV,coupe} and Price ≤ 30

CarID	Make	Model	Style	Color	Price	Mileage
C2	BMW	325	coupe	Blue	35	20
C4	Mercedes	CLK 5.0	coupe	Silver	40	35

Style not in {SUV,coupe} and Price > 30

Style in {SUV,coupe} and Price > 30

CarID	Make	Model	Style	Color	Price	Mileage
C3	BMW	745	sedan	White	45	25

PA: examples (3)

- Each class is then refined using the "2nd level" preference:

  Low(Price)

- Within the top-level class the follwoing weak order is obtained:

CarID	Make	Model	Style	Color	Price	Mileage
C5	Porsche	Cayenne	SUV	Red	25	70
C8	Nissan	350Z	coupe	Black	25	25

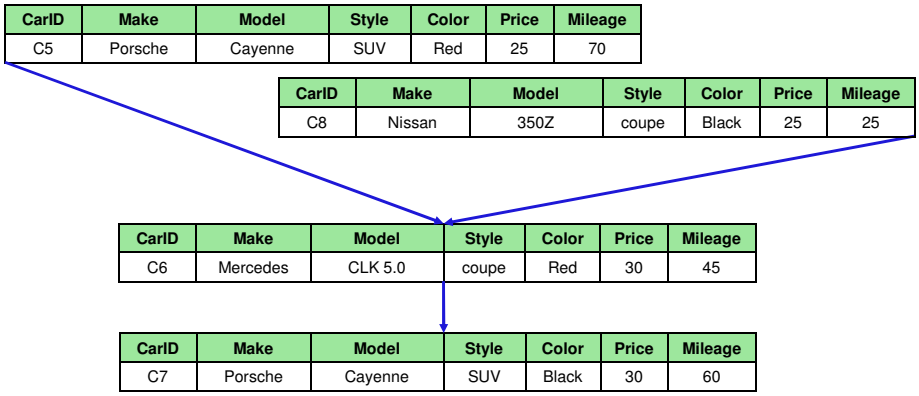
CarID	Make	Model	Style	Color	Price	Mileage
C6	Mercedes	CLK 5.0	coupe	Red	30	45
C7	Porsche	Cayenne	SUV	Black	30	60

PA: examples (4)

- The two final preferences:

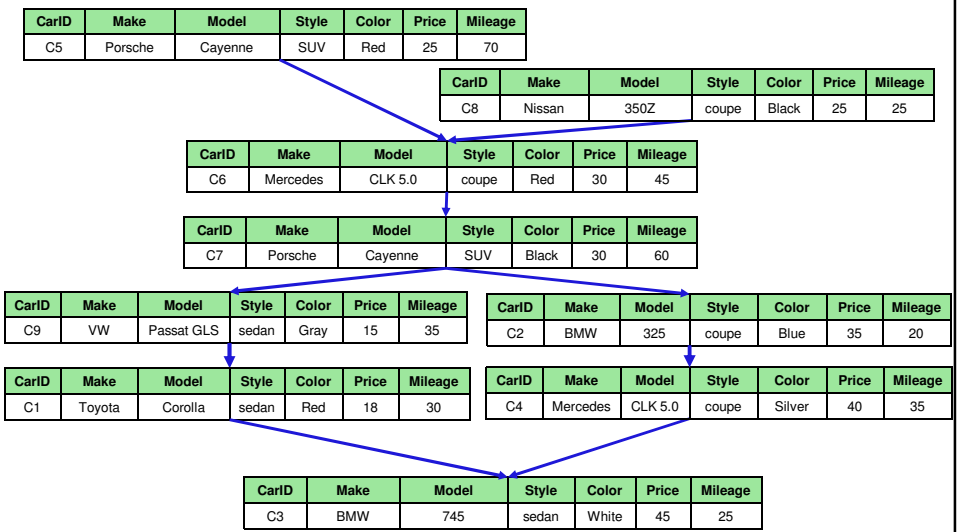
  Pos(Color='red') $\&_{SV}$ Low(Mileage)

  lead to the following (partial) preference graph:



PA: examples (5)

- The complete preference graph is:

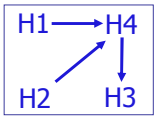


Evaluating qualitative preference queries

- The BNL algorithm can be applied whenever preferences define an spo
  - The same is true for SFS (and SaLSa), provided one is able to topologically sort the input with respect to $\succ$
- On the other hand, if $\succ$ is not transitive, BNL may fail to compute the correct result

fetched this way

Hotel	Price	Stars	Distance	Rooms
H1	30 €	2	4 km	30
H2	35 €	1	2 km	20
H4	40 €	4	6 km	50
H3	60 €	3	1 km	100



$\succ_{P,S} \otimes \succ_{D,R}$

- The BNL algorithm will work as follows:
  - Read H1: insert into the window
  - Read H2: insert into the window
  - Read H4: discard
  - Read H3: insert into the window

Result: H1, H2, and H3!?

Recap

- Although the application of qualitative preferences in DB's is a relatively new issue, it has gained increasing popularity since it is a very powerful and promising generalization of the "scores and weights" approach
- There are a number of interesting variants of the basic scenarios we have considered here, among which:
  - Conditional (including contextual) preferences
  - Preference elicitation, i.e., the process of asking the right, most effective, questions, to the user, so as to quickly narrow the search space
  - This is tightly related to the problem of designing effective user interfaces for preference specification
  - Preference revision/contraction
  - Approximate algorithms for skyline and more general preference queries
  - Skyline-based data analysis (e.g., which are the attributes that make an object part of the skyline?)
  - …