Advances in FCO-IM (2); A Shorter Algorithm for Determining Intra Fact Type Uniqueness Constraints

Jan Pieter Zwart, Guido Bakema

janpieter.zwart@han.nl, guido.bakema@han.nl
Research and Competence Group Data Architectures & Metadata Management
Academy of Communication and Information Technology
HAN University of Applied Science
Ruitenberglaan 26, 6826 CC Arnhem, The Netherlands
Phone: +31-26-3658271. Fax: +31-26-3658126

Abstract. We present a general algorithm for determining all intra fact type uniqueness constraints in a fact type with n roles. For Fact Oriented Methods of information modeling that work with elementary fact types, a top-down approach is shorter than a bottom-up approach for fact types with more than two roles. The algorithm is shorter (in terms of the number of tests to be performed) than the one published earlier in our book on FCO-IM. For its most important steps, we prove that no shorter algorithm exists.

1 Introduction

Several Fact Oriented Modeling (FOM) methods of information modeling (such as Fully Communication Oriented Information Modeling (FCO-IM) [1], Object Role Modeling (ORM) [4], Predicator Set Modeling (PSM) [6, 9] and their precursors [2, 8, 10], use the modeling construct of a fact type, which consists of a number of roles, with each role played by an object type or label type. Every fact type has at least one intra fact type uniqueness constraint (UC) on a number of its roles (possibly on all of them): the combination of values in the population of the roles to which the UC applies must be unique. An *intra fact type UC* concerns roles in only one fact type, whereas an *inter fact type UC* concerns roles in more than one fact type.

Everything stated in this paper also applies to the Relational model, if the following mapping is used: fact type ↔ table, role ↔ column, UC ↔ superkey [7, section 2.4; 4, section 4.5], UC on minimum number of roles ↔ candidate key.

Determining the UCs for all the fact types is an important step in the modeling process for all FOM methods [1, section 3.3; 4, chapter 4], not least because the structures of information models that can be derived automatically from a FOM model (such as Entity-Relationship (ER) models, Unified Modeling Language (UML) class diagrams, or Relational database schema's) depend critically upon them. The goal is to find the *smallest* UCs possible, i.e.: find UCs on a minimum number of roles (find candidate keys, not superkeys [7, section 2.4; 4, section 4.5]). The only systematic

procedure to determine UCs (just intra fact type UCs) published to date was given in [1, section 3.2.2]. Although that algorithm yields correct results, it can still be improved: here we report a shorter algorithm (in terms of the number of tests to be performed) to find all intra fact type UCs. We will consider the general problem of finding all the smallest UCs in a single fact type with n roles.

We introduce the problem using a concrete example (as always when working with FOMs) in section 2. Section 3 lists the notation we use and a few general properties of UCs. Section 4 substantiates the main strategy for the algorithm. The algorithm itself is stated and explained in section 5, which also includes an abridged version that is easier to use and covers the majority of cases. Technical details and proofs, indicated by references starting with 'C', can be found in the mathematical compendium.

2 Concrete Example

The example below briefly illustrates the relevant concepts and procedures. Figure 1 shows at the left-hand side a part of the FCO-IM fact type Class Schedule: it has five roles, a fact type expression (predicate) F1, three tuples in its example population, and two uniqueness constraints (UCs). Object types and/or label types playing the roles are not shown. From each tuple, a complete fact stating sentence can be formed by filling in all the values from the tuple in the appropriate blanks (indicated by role numbers between angle brackets) in the predicate.

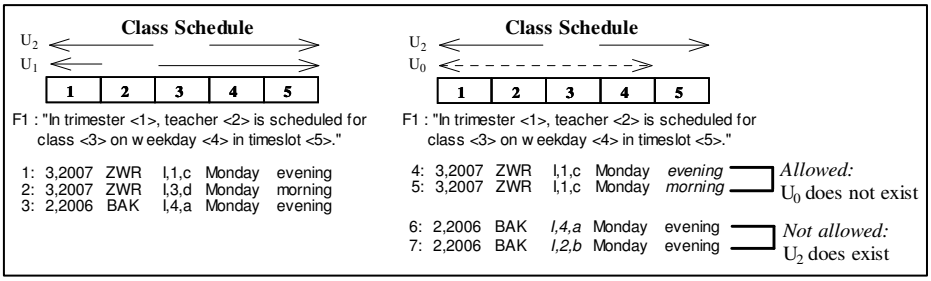


Fig. 1. Fact type Class Schedule with UCs, and how they are determined.

Fact type Class Schedule concerns the class schedules of a school. A trimester is identified by a trimester number (1, 2, 3, or 4) together with a calendar year. Classes are identified by the combination of a faculty (I stands for Information Science), a school year (1, 2, 3 or 4) and a one-letter class code. So roles 1 and 3 contain compound values, but in the context of determining UCs for fact type Class Schedule, these can be considered as atomic. See [1, section 2.11] for a further explanation.

The right-hand side of figure 1 shows how the existence of a UC is determined [1, section 3.2.2]. Tuples 4 and 5 are presented to a domain expert, who declares that they can occur together in a valid population. So U_0 does not exist: duplicate values in roles 1, 2, 3 and 4 are allowed. But the domain expert rejects tuples 6 and 7: two different classes cannot be taught by the same teacher at the same time. So U_2 does exist: duplicate values in roles 1, 2, 4 and 5 are not allowed. The problem addressed in this paper is how to systematically find all UCs on a fact type with n roles with a mini-

mum of effort on the part of both analyst and domain expert. Since each test requires exactly the same effort (create two tuples (analyst) and accept or reject their joint occurrence in a valid population (domain expert)), the number of tests to be done is a good measure of the length of UC-determining algorithms.

In FCO-IM, fact types with more than 4 roles are actually very rare: fact types with 2 roles are most common (≥ 80 %), fact types with 3 roles occur frequently (≤ 20%) , fact types with 4 roles occasionally (≤ 1%), and we have only once encountered a fact type with 5 roles in practice. Usually combinations of values become clustered as complex identifiers of object types, as in the compound values in roles 1 and 3. In a large FCO-IM information model, roles 1, 4 and 5 would probably be combined in an object type Lesson Period, reducing Class Schedule to a fact type with just three roles. Still, minimizing the work needed to determine UCs even for fact types with only three roles is relevant in view of their importance.

3 Notation and properties of UCs

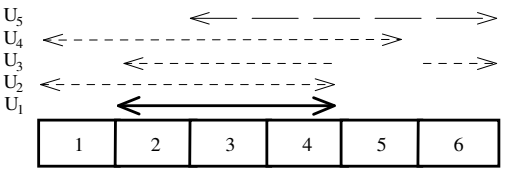
We use the following notation:

U_k	UC number k.
$U_k(p;n)$	UC number k on p roles in a fact type with n roles ($1 \le p \le n$).
$R(U_k)$	the collection of roles U_k operates on.
$Pop(R(U_k))$	the population of the collection of roles on which U_k operates.
$\#R(U_k)$	the number of roles in $R(U_k)$.
$\#UC(p;n)$	the number of UCs on p roles in a FT with n roles ($1 \le p \le n$).

The following properties of uniqueness constraints (UCs) hold:

P1 If there exists a $U_k(p;n)$, then each tuple in $Pop(R(U_k(p;n)))$ is unique.
This is just the definition of a UC.

P2 Suppose there exists a certain $U_k(p;n)$ on $R(U_k)$, with $1 \le p < n$.
Then there exist also all UCs $U_l(q;n)$ on $R(U_l(q;n))$ with $p < q \le n$,
and $R(U_k(p;n))$ a subset of $R(U_l(q;n))$.
Example: In the fact type with six roles shown here, U_1 (with $R(U_1)=\{2, 3, 4\}$), implies the existence of U_2, U_3, U_4, and four other UCs not shown, but not U_5.



Proof: This follows directly from P1: if every tuple in the population of $R(U_k)$ is unique, then certainly every tuple in the population of every set of roles that completely contains $R(U_k)$ is also unique.

P3 Every fact type has at least one UC.
This is a meta-constraint on FOM models: two identical tuples are forbidden in all valid populations of any fact type. Otherwise we would allow redundancy by sheer repetition of facts.
Remark: it is possible to adopt the convention that if there is only one UC on all n roles of a fact type, then it is not necessary to explicitly draw it in. For practical reasons we prefer not to use this convention. One reason is that we can easily visually tell the difference between FTs for which we have already

finished determining the UCs, and FTs for which we have not yet done so (otherwise absence of evidence gets confused with evidence of absence).

4 Main strategy

4.1 Elementary fact types and n-1 rule

Most FOM methods model *elementary* facts. Since it is not completely clear what an elementary fact exactly is [1, section 2.3; 4, section 3.3; 5], we will use the following working definition in this paper:

A fact type (FT) with n roles is elementary if and only if it cannot be replaced by two or more FTs with less than n roles without losing information. Losing information means: there exists a population of the original FT with n roles, which cannot be reconstructed exactly by joining the populations of the FTs with less than n roles.

From this follows the well-known n-1 rule [1, section 3.3.1.1; 4: section 4.5]: a FT with n roles is not elementary if there is at least one uniqueness constraint (UC) on less than n-1 roles. Proof: Suppose there is a fact type FT1 with n roles that has a UC on the n-2 roles {1, 2, … n-2}. Then FT1 can be split into two fact types FT1a and FT1b with n-1 roles: FT1a has roles {1, 2, …, n-2, n-1}, and FT1b has roles {1, 2, …, n-2, n}, and both have a UC on roles {1, 2, …, n-2}. Joining the populations of FT1a and FT1b will always exactly regenerate the population of FT1, because roles n-1 and n are both functionally dependent on roles {1, 2, … n-2}. Finally: If FT1 has a UC on less than n-2 roles, then it also has a UC on n-2 roles (property P2 in section 3).

4.2 Top-down versus bottom-up

We can determine all UCs on a fact type (FT) either bottom-up or top-down.

Bottom-up: first test whether there are any UCs on 1 role, then on 2 roles, and so on. If none are found on 1, 2, ..., n-1 roles, then there must be one UC on all n roles (otherwise we would have exactly the same fact more than once in the population).

Top-down: first test whether there are any UCs on n-1 roles. If not, then there must be one UC on all n roles. If there is at least one UC on n-1 roles, then we must check if there are any UCs on n-2 roles (because of property P2 in section 3). If so, then we must check for UCs on n-3 roles, and so on until we find no smaller UCs on n-p roles, for some p with $1 < p < n$.

Which approach is more efficient? The bottom-up approach requires at least n tests (all possible UCs on 1 role, from C2e with m=1, and C1a), and at most 2^n-2 tests (from C2d, C2c, C1b and the fact we don't have to test UCs on 0 roles or on n roles), whereas the top-down approach also requires at least n tests (all possible UCs on n-1 roles, from C2e with m=n-1, and C1a), and at most 2^n-2 tests (same reason as for the bottom-up approach). However, FCO-IM works with elementary FTs, and practice shows that even inexperienced modelers will only very rarely ($\leq 1\%$) model a non-elementary FT by mistake. It is therefore unlikely that any UC on less that n-1 roles exists. In the ideal situation (if we know there are no non-elementary fact types) the

following number of tests have to be carried out for a FT with n roles:

n	2	3	4	5	6	…
Top-down (n)	2	3	4	5	6	…
Bottom-up (2^n-2)	2	6	14	30	62	…

The top-down approach is clearly much more efficient for fact types with 3 or more roles, and equally efficient for fact types with 2 roles. So we will consider only the top-down approach further. We can estimate that even in the non-ideal situation (in which we will have to check for non-elementariness), the average number of tests for n=3 will be ≈ 3.5 for top-down versus ≈ 5.5 for bottom-up.

5 Algorithms

5.1 General algorithm for finding all intra fact type UCs in a FT with n roles.

In the algorithm given below, 'test a UC' means: determine whether a UC exists by presenting two tuples with suitably chosen values to a domain expert, as the right side of figure 1 shows. The algorithm is given in pseudo code. Details are explained below the algorithm itself.

1 n = 1.
  Add a UC on the single role in the fact type.
2 n > 1.
  a Test all the n UCs on n-1 roles. Suppose m are found ($0 \leq m \leq n$). Add all found UCs to the fact type.
  b IF m = 0 THEN there is a UC on all n roles. Add it to the fact type.
  c Set p=1.
  d IF m < p+1 OR p = n-1
    i THEN the analysis is complete.
    ii ELSE
      (1) FOR all ½m(m-1) pairs of UCs U_i, U_j, $1 \leq i < j \leq m$, on n-p roles found do:
        (a) IF #($R(U_i) \cap R(U_j)$) = n-(p+1)
          (i) THEN IF the same overlap has not yet been tested
                THEN test UC U_k on $R(U_i) \cap R(U_j)$
              ENDIF.
        ENDIF
      ENDFOR
      (2) FOR all UCs found in 2dii(1) do:
        (a) Replace U_i and U_j by U_k (this may concern several pairs U_i, U_j with the same overlap for one U_k).
      ENDFOR
      (3) Set m equal to the number of UCs found in step 2dii(1).
      (4) Increase p by 1 (set p = p+1).
      (5) GO TO 2d.
    ENDIF

Explanation:

Step 1: Given property P3 in section 3, a UC on the single role is the only possibility. See also the remark made in property P3.

Steps 2a-b: All n UCs on n-1 roles must be tested. If none is found, then there is only one UC on all n roles (from property P3 and C7b). Otherwise we need to check if there are any UCs on less than n-1 roles.

Step 2c: Sets the initial value for p, which increases by 1 in each recursive step. In each recursive step, UCs on n-(p+1) roles are determined.

Step 2d: Criteria to stop the recursion. The algorithm terminates if p=n-1, because there can be no UCs on n-(n-1+1)=0 roles. It also stops if less than p+1 UCs on n-p roles are found, because no UCs on less than n-p roles can then exist (proof: see C7e).

Step 2dii(1): A UC on n-(p+1) roles can only occur if there are two UCs on n-p roles with an overlap of n-(p+1) roles (proof: see C8c). Therefore all pairs of UCs on n-p roles must be considered.

Step 2dii(1)(a): A test is carried out only if the overlap in the pair is n-(p+1). For p=1, all UCs have an overlap of n-2 roles, but for p>1 this is not necessarily so.

Step 2dii(1)(a)(i): For p=1, no two pairs have the same overlap, but for p>1 several pairs can have the same overlap. In such cases only one test is to be done for all pairs with the same overlap. Perhaps some criteria could be given to find collections of pairs with the same overlap easily, which would further shorten the algorithm, but we have not found them.

Step 2dii(2): Since smaller UCs imply larger UCs, we replace the larger by the smaller. Each UC on n-(p+1) roles replaces all UCs on n-p roles that have these n-(p+1) roles as a subset. If more than one pair considered in step 2dii(1) have the same overlap, then more than two UCs on n-p roles are replaced.

Steps 2dii(3)-(5): m and p are updated, and the next recursive step to find yet smaller UCs is taken.

5.2 Shortness of the algorithm.

In step 2a, all n UCs on n-1 roles are tested. This step obviously cannot be shortened, since each UC can either be present or not.

In step 2dii(1) for p=1, the overlap in all pairs of UCs found on n-1 roles must be tested. This calls for ½m(m-1) = m-over-2 tests. We show in C7e: if #UC(n-1;n) = m, then $0 \leq$ #UC(n-2;n) $\leq$ m-over-2, and m is the smallest integer for which this is true. So once more the number of tests to be carried out is equal to the number of UCs possible, and each UC can either be present or not. For these two steps the algorithm is therefore the shortest possible. However, we have not proved the same for $p > 1$. We expect that in over 90% of all cases in practice the algorithm will stop after these two steps, so it is quite efficient.

The algorithm is shorter (in terms of number of tests to be done) than the one presented in [1, section 3.2.2], which calls for more tests because it has less strict termination conditions (for instance, it calls for checks even if only one UC on n-1 roles is found, but its recursion is formulated in a simpler way). Even so it will give the same results as the algorithm presented here, which takes less tests.

5.3 Abridged algorithm for finding intra-fact type UCs in a FT with n roles.

The general algorithm given above finds all the smallest UCs (candidate keys) in a fact type with n roles. Since FOM methods usually work with elementary fact types, the algorithm will terminate very quickly. Below, we give an abridged version that is easier to use and covers over 99% of the cases encountered in FCO-IM modeling. It is phrased less abstractly, and stops as soon as a violation of the n-1 rule is found, indicating that the general algorithm can be followed further if desired, and what the consequences are if it is not.

1 Fact type with 1 role.
  Add a UC on the single role in the fact type.
2 Fact type with more than one role (n > 1).
  Test all the n UCs on n-1 roles.
  a No UC on n-1 roles is found. Then there is one UC on all the n roles.
  b At least one UC on n-1 roles is found.
    i The fact type has 2 roles. The analysis is complete.
    ii The fact type has 3 or more roles.
      (1) Only one UC on n-1 roles is found. The analysis is complete.
      (2) More than one UC on n-1 roles is found, say m, with $2 \leq m \leq n$.
        For each of the ½m(m-1) pairs of UCs on n-1 roles, test the UC on the n-2 roles in the overlap of the two UCs in the pair.
        (a) No UC on n-2 roles is found. Then all the UCs on n-1 roles are correct and the analysis is complete.
        (b) At least one UC on n-2 roles does exist. Then each UC on n-2 roles replaces the pair of UCs on n-1 roles from which it was found. The fact type is not elementary and should be remodeled by splitting it into two or more smaller fact types. For fact types with 3 roles, the analysis is complete, and splitting should be performed. For fact types with more than three roles: if the UC-analysis is not continued from here, then there is a small risk of missing UCs on the fact type with n roles that will become inter fact type UCs after splitting. The general algorithm can be followed further if desired, If not, split the fact type, and do the UC analysis for the new fact types.

6 Conclusion

The algorithms presented in this paper minimize the work to be done when determining all the smallest intra fact type uniqueness constraints (UCs) in elementary fact types (i.e.: find all the 'candidate keys' of elementary fact types). This is important in practice, because the structure of any logical data model (in ERM, UML, Relational or other technique), which nowadays can be derived automatically from the conceptual elementary model, depends critically on these UCs. Apart from saving time and effort, the fewer tests there are to be carried out, the smaller the chances are of making mistakes, which will result in faulty structures.

Mathematical compendium

C1 Binomial coefficients for integers n, m: definitions.

a): $\binom{n}{m} = \frac{n!}{(n-m)!m!}$ b): $\binom{n}{0} = \binom{n}{n} = 1$ c): $\binom{n}{m} = 0$ for $m < 0$ and $m > n$.

d): $\binom{n}{m}$ is pronounced as 'n – over – m'.

C2 Well-known properties of binomial coefficients.

a): $\binom{n}{2} = \frac{1}{2}n(n-1)$ b): $\binom{n}{m} = \binom{n}{n-m}$ c): $\sum_{i=0}^{n}\binom{n}{i} = 2^n$

d): n-over-m is the number of different subsets with m elements that can be formed from a set of n elements.

e): n-over-m is the number of possible UCs on m roles in a FT with n roles.
Proof: From C2d, with roles as elements and each UC on a different subset.

C3 Smallest integer greater than.

$\lceil x \rceil :=$ smallest integer greater than x. $\lceil 2.9 \rceil = 3$, $\lceil 3 \rceil = 4$, $\lceil 3.1 \rceil = 4$

C4 Relationships between #UC(n-p;n), #UC(n-p+1;n) and #UC(n-1;n).

a): $\#UC(n-p;n) = 1 \Rightarrow \#UC(n-1;n) = \binom{p}{p-1} = p$, for $1 \le p < n$

Proof: with p roles not under the given UC, the number of ways in which the given UC can be extended to cover n-1 roles is the number of ways to choose p-1 roles out of p roles (C2d).

b): $\#UC(n-p;n) = 1 \Rightarrow \#UC(n-p+1;n) = \binom{p}{1} = p$, for $1 \le p < n$

Proof: with p roles not under the given UC, the number of ways in which the given UC can be extended to cover p+1 roles is the number of ways to choose 1 role out of p roles (C2d).

c): $\#UC(n-1;n) < p \Rightarrow \#UC(n-p) = 0$

Proof: From the negation of C4a with its '='-signs replaced by '≥'-signs.

C5 Relationships between #UC(n-q;n) and #UC(n-p;n) roles for p < q

a) $\#UC(n-q;n) \ge 1 \Rightarrow \#UC(n-p;n) \ge 1$, for $0 \le p < q < n$.

Proof: Follows directly from property P2.

b) $\#UC(n-p;n) = 0 \Rightarrow \#UC(n-q;n) = 0$, for $1 \le p < q < n$.

Proof: From the negation of C5a, with '< 1' replaced by '= 0'.

c) $\#UC(n-(p+1);n) \geq 1 \quad \Rightarrow \quad \#UC(n-p;n) \geq p+1 \quad , \quad \text{for } 0 \leq p < n.$

Proof: Follows directly from C4b.

d) $\#UC(n-p;n) < p+1 \quad \Rightarrow \quad \#UC(n-(p+1);n) = 0 \quad , \quad \text{for } 1 \leq p < n.$

Proof: From the negation of C5c, with '< 1' replaced by '= 0'.

e) $\#UC(n-p;n) < p+1 \quad \Rightarrow \quad \#UC(n-q;n) = 0 \quad , \quad \text{for } 1 \leq p < q < n.$

Proof: From C5d and C5b.

C6 Relationships between #UC(n-(p+1);n) and #UC(n-p;n):

a): If there is a UC $U_{k,o}$ on n-(p+1) roles, then there are p+1 UCs $U_{k,1}$, …, $U_{k,p+1}$ on n-p roles, for $0 \leq p \leq n-2$. For each $U_{k,i}$: $R(U_{k,0}) \cap R(U_{k,i}) = R(U_{k,0})$.
Proof: Follows directly from C4b and P2.

b): If $p > 0$ then for each pair $U_{k,i}$, $U_{k,j}$ from C6a: $R(U_{k,i}) \cap R(U_{k,j}) = R(U_{k,0})$.
Proof: From C6a with $p > 0$, and the fact that $U_{k,i} \neq U_{k,j}$ for $i \neq j$.

c): If there are 2 UCs $U_{k,i}$ and $U_{k,j}$ on n-p roles and $\#(R(U_{k,i}) \cap R(U_{k,j})) < n-(p+1)$, then there is no UC on n-(p+1) roles from which they both follow.
Proof: From the negation of C6b, and if $\#(R(U_{k,i}) \cap R(U_{k,j})) \neq R(U_{k,0})$ then it must be smaller than $R(U_{k,0})$ because $U_{k,i} \neq U_{k,j}$ for $i \neq j$.

C7 Bounds on #UC(n-p;n) given #UC(n-1;n) and vice versa.

a): $\#UC(n-p;n) = k \quad \Rightarrow \quad \#UC(n-1;n) \leq \min(n, pk) \quad , \quad \text{for } 1 \leq p < n \quad , \quad 1 < k \leq \binom{n}{n-p}$

Proof: From C4a, and choosing the UCs on n-p roles so that as many as possible of the remaining p roles get covered by the UCs on n-1 roles.

b): $\#UC(n-p;n) \geq \binom{m-1}{p} + 1 \quad \Rightarrow \quad \#UC(n-1;n) \geq m \quad , \quad \text{for } 1 \leq p < n, \quad 1 \leq m \leq n$

Proof: One UC on n-p roles generates p UCs on n-1 roles (C4a). Suppose we have m UCs on n-1 roles, with p<m<n. What is the greatest number of UCs on n-p roles that can generate these m UCs on n-1 roles? This is equivalent to the question how many different subsets of p UCs on n-1 roles can be formed from the given set of m UCs, because each subset corresponds to one UC on n-p roles, and no two UCs on n-p roles can give the same subset. So (from C2d): at least m-over-p UCs on p roles are needed to generate at least m+1 UCs on n-1 roles. The inequality now follows by replacing m by m-1 in the sentence above.

c): $\#UC(n-p;n) = k \text{ and } m \text{ defined by } \binom{m-1}{p} \leq k-1 < \binom{m}{p} \quad \Rightarrow$

$m \leq \#UC(n-1;n) \leq \min(n, pk) \qquad \text{for } 1 \leq p < n, \quad 1 < k \leq \binom{n}{n-p}$

Proof: From C7b, C7c, and C7c again with m replaced by m+1.

d): $\#UC(n-2;n)=k \Rightarrow \left\lceil \frac{1}{2}\left(1+\sqrt{1+8(k-1)}\right)\right\rceil \leq \#UC(n-1;n) \leq \min(n, 2k)$, for $1 \leq k \leq \frac{1}{2}n(n-1)$

See C3 for the definition of the brackets '⌈' and '⌉' used above.
Proof: From C7c with p=2, expanding the binomial coefficients to obtain the quadratic inequality $(m-1)(m-2) \leq 2(k-1) \leq m(m-1)$ and solving for m.

e): $\#UC(n-1;n)=m \quad \Rightarrow \quad 0 \leq \#UC(n-2;n) \leq \binom{m}{2}$, for $1 \leq m \leq n$

Proof: Left inequality: obviously the existence of a UC on n-1 roles does not imply the existence of any UC on less than n-1 roles.
Right inequality: For m = 1 this follows directly from C4c with p = 2, and from C1c. For m > 1: observe that $\frac{1}{2}(1+\sqrt{(1+8(k-1))})$ is the positive root of the equation $m^2-m-2(k-1)=0$. From C7d: $m > \frac{1}{2}(1+\sqrt{(1+8(k-1))})$, which implies $m^2-m-2(k-1) > 0$. So $\frac{1}{2}m(m-1) \geq k$. From C2a the inequality follows.

References

1 Bakema, Guido; Zwart, Jan Pieter; Lek, Harm van der: Fully Communication Oriented Information Modeling (FCO-IM), 2002. The book can be downloaded for free from the website of [3]: http://www.casetalk.com/php/index.php?FCO-IM%20English%20Book.
2 Bakema, G.P.; Zwart, J.P.C; Lek, H. van der: Fully Communication Oriented NIAM, NIAM-ISDM 1994 Conference, Working Papers, Albuquerque, USA (1994).
3 BCP Software: CaseTalk, FCO-IM modeling tool, see http://www.CaseTalk.com.
4 Halpin, Terry: Information Modeling and Relational Databases, Morgan Kaufmann Publishers, 2001, ISBN-13: 978-1-55860-672-2, ISBN-10:1-55860672-6.
5 Halpin, Terry: What is an elementary fact?, NIAM-ISDM 1993 Conference, Working Papers, Albuquerque, USA (1993). Also available in slightly edited form on the web from http://www.orm.net/pdf/ElemFact.pdf.
6 Hofstede ter, Arthur H.M: Information Modelling in Data Intensive Domains, PhD-thesis (1993), Radboud University Nijmegen, the Netherlands, ISBN 90-9006263-X.
7 Korth, Henry K, Silberschatz, Abraham: Database System Concepts, McGraw-Hill Book Company, 1986, ISBN 0-07-044752.
8 Nijssen, G.M.; Halpin, T.A.: Conceptual Schema and Relational Database Design, Prentice Hall, 1989, ISBN 0-7248-0151-0.
9 Proper, H.A: A theory for Conceptial Modelling of Evolving Application Domains, PhD Thesis (1994), Radboud University, Nijmegen, the Netherlands, ISBN 90-9006849-X.
10 Wintraecken, J.J.V.R.: Informatie-analyse volgens NIAM, Academic Service, 1985, ISBN 90-6233-169-6.