Matching and Merging of Ontologies Using Conceptual Graphs

Gopinath Ganapathy, Ravi Lourdusamy

***Abstract*—Knowledge management applications need to determine whether two or more knowledge representations encode the same knowledge. Solving this *matching problem* is hard because representations may encode the same content but differ substantially in form. Previous approaches to this problem have used either syntactic measure or semantic knowledge to determine the distance between two representations. The aim of this article is to define matching of m conceptual graphs and present the mathematical aspects of matching and binding of m conceptual graphs with fitness. The algorithms developed for matching and merging of m conceptual graphs are illustrated.**

***Index Terms*—Binding, Taxonomy, Transformation, Fitness.**

I. Introduction

A requirement common to many knowledge applications is to determine whether two or many knowledge representations, encoded using the same ontology, capture the same knowledge. The task of determining whether two or more representations encode the same knowledge is treated as a graph matching problem. The knowledge representation is encoded using conceptual graph. The representations capture the same knowledge if their corresponding graphs match. The multiple encoding of the same knowledge rarely match exactly, so a matcher must be flexible to avoid a high rate of false-negatives. However, a matcher that is too flexible can suffer from a high rate of false-positives [6]. This problem has various causes, including

(i) The ontology is expressible enough to allow the same information to be encoded in different ways
(ii) The representations are built by different knowledge engineers (or computer programs), raising the likelihood they differ
(iii) The representations are large, increasing the opportunity for differences.

Previous solutions to this problem have produced two types of matchers. *Syntactic matchers* use only the graphical form of the representations, judging their similarity by the amount of common structures shared [1], [2] or the number of edit operations required to transform one graph into the other [4], [7], [8], [10]. Approaches that focus on the amount of shared common structures do not handle mismatches. Approaches that use edit operations can handle mismatches but are sensitive to the cost assigned to the edit operations and tuning these parameters optimally is problematic.

In contrast, *semantic matchers* use knowledge, stored in an ontology, of the terms referenced in the representations. Semantic matchers use this knowledge to determine the match of two representations. [3], [5], [6], [12]. The knowledge encoded can be equivalently encoded as conceptual graphs [9]. The algorithm for matching of two graphs is discussed in [11]. In this paper the generalized algorithm for matching of m conceptual graphs is developed. The fitness of the matched graph is further investigated. Further the algorithm is extended to merge $m-$conceptual graphs.

The article is organized as follows: Section 1 deals with an introduction. The terminologies involved are outlined in Section 2. In Section 3, the algorithm for matching $m-$ conceptual graphs is illustrated. The algorithm for merging $m-$ conceptual graphs is illustrated in Section 4 and in Section 5 conclusion is presented.

Date of paper submission is March 6th 2011.

Gopinath Ganapathy is with the Department of Computer Science, Bharathidasan University, Trichy, India. (e-mail: gganapathy@gmail.com).
Ravi Lourdhusamy is with Department of Computer Science, Sacred Heart College, Tirupattur, India. (+91 9443280319; fax: +91 4179 225060; e-mail: raviatshc@yahoo.com).

II. Preliminaries

In this section, we present some definitions and preliminaries which will be useful for further discussion.

Definition 2.1: [6] Transitive and part ascendant transformations conform to a more general notion called 'transfers through'. A relation r transfers through another relation r' if

$$X \xrightarrow{r} Y \xrightarrow{r'} Z \Longrightarrow X \xrightarrow{r} Z \tag{1}$$

Definition 2.2: [6] A triple is a 3-tuple of the form $(head, relation, tail)$ where head and tail are concepts or instances (i.e., nodes in a conceptual graph) and relation is an edge in the graph. Every two nodes connected by an edge in a conceptual graph can be mechanically converted into a triple and hence a conceptual graph into a set of triples.

Definition 2.3: The n triples $t_1 = (head_1,\ relation_1,\ tail_1)$, $t_2 = (head_2, relation_2, tail_2), \ldots,$ t_n = $(head_n,\ relation_n, tail_n)$ of graph G align if $head_1 \geq head_2 \geq \ldots \geq head_n$, $uneg(relation_1) \geq uneg(relation_2) \geq \ldots \geq uneg(relation_n)$ and $tail_1 \geq tail_2 \geq \ldots \geq tail_n$. The uneg(relation) unnegates relation if it is negated otherwise returns the relation.

Definition 2.4: For ℓ= $\{(t_{11},\ t_{21},\ \ldots\ t_{m1}),\ (t_{12},\ t_{22},\ \ldots\ t_{m2}), \ldots, (t_{1n_1},\ t_{2n_2}, \ldots, t_{mn_m})\}$, a list of aligned triples of m graphs , the bindings for ℓ ie.,

$$b(\ell) = \{(head_{11}/head_{21}/\ldots/head_{m1}, tail_{11}/tail_{21}/\ldots/tail_{m1}),$$
$$(head_{12}/head_{22}/\ldots/head_{m2},\ \ tail_{12}$$
$$/tail_{22}/\ldots/tail_{m2}), \ldots, (head_{1n_1}/head_{2n_2}/\ldots/head_{mn_m},$$
$$tail_{1n_1}/tail_{2n_2}/\ldots/tail_{mn_m})\}.$$

III. Matching of m- conceptual graphs

A. Binding of $m-$ conceptual Graphs

Given m graphs $G_1 = \{t_{11}, t_{12}, \ldots t_{1n_1}\}$, $G_2 = \{t_{21}, t_{22}, \ldots, t_{2n_2}\}, \ldots, G_m = \{t_{m1}, t_{m2}, \ldots, t_{mn_m}\}$ where $n_1, n_2, \ldots, n_m$ are the number of triples of G_1, G_2,...,G_m respectively and a set of r transformations R where $R = \{R_1, R_2, \ldots, R_r\}$. The aim of the algorithm is to find a common subgraph of G_1, G_2,... and G_m called SG. Construct a list M of all possible alignments between the triples of G_1, G_2,... and G_m. Each element of M is of the form $\ell = \{(t_{11}, t_{21}, ..., t_{m1}),\ (t_{12}, t_{22}, ..., t_{m2}),$ $...(t_{1n_1}, t_{2n_2}, ..., t_{mn_m})\}$. The generalized algorithm for finding a match between m representations, is presented as Algorithm-1. The steps for finding a match between m representations is illustrated with an example of three graphs, G_1, G_2 and G_3 generated by organization structure of three hospitals shown in $Figures - 1, 2$ and 3. For reference, we label each triple in G_1 with a unique number from 1 to 22, each triple in G_2 with a unique upper case letter from A to V and each triple in G_3 with a unique lowercase letter from a to z and from aa to hh. We use subscripts to differentiate terms that appear multiple times (e.g., Hospital$_-$Model$_-$1).

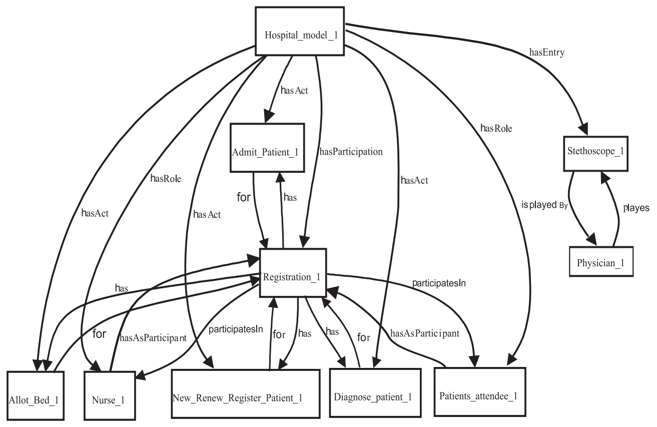

Figure - 1 Graph G_1

Algorithm - 1: Outline of the generalized matching algorithm

```
1) $M = NIL$ and $\ell = NIL$
   $FOR$ each triple $t_{1i_1}$ in $G_1$
      $FOR$ each triple $t_{2i_2}$ in $G2$
         ....................................
            $FOR$ each triple $t_{mi_m}$ in $G_m$
                  If $t_{1i_1}, t_{2i_2}, ..., t_{mi_m}$are aligned
                  $THEN$ add $(t_{1i_1}, t_{2i_2}, ..., t_{mi_m})$ to $\ell$.
      $ADD$ $\ell$ to $M$ and reset $\ell$ to $NIL$.
2) Use $M$ to construct a common subgraph of $G_1$, $G_2$,..., $G_m$
   called $SG$.
   $SG = \{(t_{11}, ..., t_{m1}), (t_{12}, ..., t_{m2}), ..., (t_{1n_1}, ..., t_{mn_m})\}$
   where $(t_{1i}, ..., t_{mi})$ are the aligned triples of $G_1$, $G_2$, ..., $G_m$
   respectively.
3) $IF$ $SG$ is inconsistent $THEN$ stop and return $NIL$
4) $FOR$ each rule $R_i$ in $R$,
   $FOR$ each $j = 1, 2, ..., m$
      $FOR$ each $k = 1, 2, ..., m$
         $IF$ $R_i$ is applicable to $G_j$ with respect to $G_k$
         $THEN$ apply $(R_i, G_j, G_k)$
5) $FOR$ each $j = 1, 2, ..., m$
   $FOR$ each unaligned triple $t_{ji_j}$ in $G_j$
      $IF$ $t_{1i_{1_j}}, t_{2i_{2_j}}, ..., t_{mi_{m_j}}$ are aligned and
         $b(\{(t_{1i_{1_j}}, t_{2i_{2_j}}, ..., t_{mi_{m_j}})\})$ is consistent with $b(SG)$,
      $THEN$ add$(t_{1i_{1_j}}, t_{2i_{2_j}}, ..., t_{mi_{m_j}})$ to $SG$ and break.
   $UNTIL$ $SG$ reaches quiescence go to step 4.
6) $RETURN$ $SG$
```

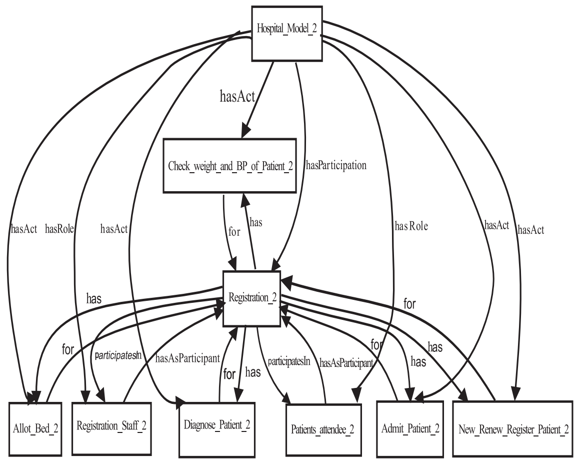

Figure - 2 Graph G_2

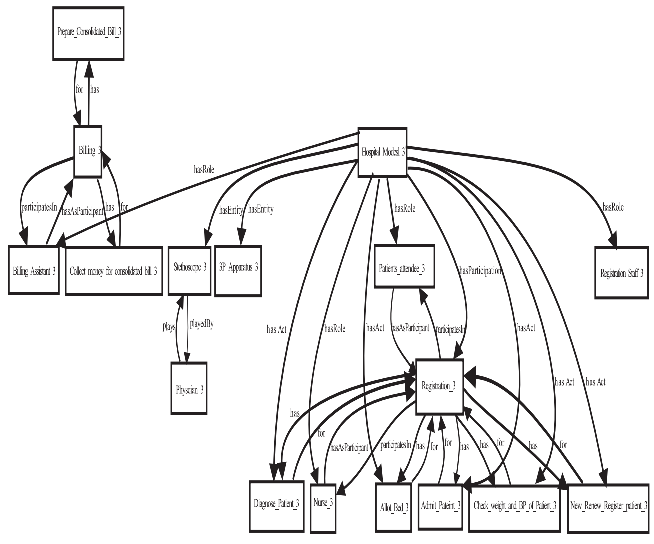

Figure - 3 Graph G_3

In step 1 the generalized algorithm compares each triple in G_1 with each triple in G_2, G_3,...,G_m to find all possible alignments. In our example t_{11} aligns with t_{21} and t_{31}. Triple t_{11}, however, does not align with triple t_{22} and its combinations because the relations differ. The initial match is denoted by M. $M = \{\ell_1, \ell_2, ..., \ell_p\}$ where p is the total number of possible alignments. Each element of M is a list called ℓ_i. For example $\{(t_{11}, t_{21}, t_{31})\}$ is called ℓ_1, $\{(t_{12}, t_{22}, t_{32})\}$ is called ℓ_2, etc. In step 2 the generalized algorithm uses M to construct common subgraph of G_1, G_2,...,G_m called SG. The generalized algorithm begins by selecting a member, ℓ_i, of M to serve as the seed of the construction process (recall that $M = \{\ell_1, \ell_2, ..., \ell_s\}$) and $\ell_i = \{(t_{1i_{11}}, t_{2i_{21}}, t_{mi_{m1}}), (t_{1i_{12}}, t_{2i_{22}}, t_{mi_{m2}}), ...,$ $(t_{1i_{1k_i}}, t_{2i_{2k_i}}, t_{mi_{mk_i}})\}$, $i = 1, 2, ..., s$. This seed is selected based on a heuristic scoring function

$$h(\ell_i) = \frac{1}{k_i}\left\{\sum_{j=1}^{k_i} n(head_{1i_{1j}}/head_{2i_{2j}}/.../head_{mi_{mj}})\right.$$
$$\left. + n(tail_{1i_{1j}}/tail_{2i_{2j}}/..../tail_{mi_{mj}})\right\} \qquad (2)$$

where $head_{1i_{1j}}/head_{2i_{2j}}/, ..., /head_{mi_{mj}}$ and $tail_{1i_{1j}}/tail_{2i_{2j}}/, ..., /tail_{mi_{mj}}$ are the bindings of $t_{1i_{1_j}}, t_{2i_{2_j}}, ..., t_{mi_{m_j}}$ and $n(b)$ is the number of times the bindings b occurs in $binding(M)$. This function h is a heuristic that favors those ℓ_i in M with interconnectivity. Bindings that occur frequently indicate high interconnectivity. We want to select these ℓ_i as the seeds because they have more potential for allowing larger common subgraphs to be constructed. Therefore, the algorithm selects the ℓ_i in M with the highest score, as determined by the function h. SG is extended with those pairs of aligned triples in M whose bindings intersect the binding of the pairs in SG. Pairs in M that extend SG are removed from M along with ℓ_j they belong to. This process is repeated until SG can no longer be extended.

In steps 3-5, the generalized algorithm checks if SG is consistent. SG is inconsistent if it contains an aligned m-tuples of triples $(t_{1i}, t_{2i}, ..., t_{mi})$ where the relation of at least one t_{ji} is negated and the relation of at least one t_{ki} is not negated. If SG contains such a m-tuples, then generalized algorithm stops and returns NIL. Otherwise, the generalized algorithm applies transformations to improve the match(i.e., steps 4 and 5). Steps 4 and 5 are repeated until SG reaches quiescence. In step 4, the generalized algorithm applies transformations to resolve mismatches among $G_1, G_2, ..., G_m$. In step 5, the generalized algorithm will try to align additional triples among $G_1, G_2, ..., G_m$. Step 5 is like step 1 except that it focuses on the unaligned triples. We have identified a set of transformations for the health care domain. These transformations are used to improve the matching of m-conceptual graphs in the domain. Returning to our example, the triples of G_1, G_2 and G_3 are as follows:

G_1 = {(Hospital_Model_1, hasAct, Allot_Bed_1), (Hospital_Model_1, hasAct, New_Renew_Register_patient_1), (Hospital_Model_1, hasAct, Admit_patient_1), (Hospital_Model_1, hasParticipation, Registration_1), (Hospital_Model_1, hasAct, Diagnose_patient_1), (Hospital_Model_1, hasRole, Patients_attendee_1), (Admit_patient_1, for, Registration_1), (Registration_1, has, Admit_patient_1), (Registration_1, has, Allot_Bed_1), (Allot_Bed_1, for, Registration_1) (New_Renew_Register_patient_1, for, Registration_1), (Registration_1, has, New_Renew_Register_patient_1), (Registration_1, has, Diagnose_patient_1), (Diagnose_patient_1, for, Registration_1), (Patients_attendee_1, hasAsParticipant, Registration_1), (Registration_1, participatesIn, Patients_attendee_1), (Hospital_Model_1, hasRole, Nurse_1), (Hospital_Model_1, hasEntity, Stethoscope_1), (Stethoscope_1, isPlayedBy, Physician_1), (Physician_1, plays, Stethoscope_1), (Nurse_1, hasAsParticipant, Registration_1), (Registration_1, participatesIn, Nurse_1)}

G_2={(Hospital_Model_2, hasAct, Allot_Bed_2), (Hospital_Model_2, hasAct, New_Renew_ Registration_ patient_2), (Hospital_Model_2, hasAct, Admit_patient_2), (Hospital_Model_2, hasParticipation, Registration_2), (Hospital_Model_2, hasAct, Diagnose_patient_2), (Hospital_Model_2, hasRole, Patients_attendee_2), (Admit_patient_2, for, Registration_2), (Registration_2, has, Admit_patient_2), (Registration_2, has, Allot_Bed_2), (Allot_Bed_2, for, Registration_2), (New_Renew_Register_patient_2, for, Registration_2), (Registration_2, has, New_Renew_Register_patient_2), (Registration_2, has, Diagnose_patient_2), (Diagnose_patient_2, for, Registration_2), (Patients_attendee_2, hasAsParticipant, Registration_2), (Registration_2, participatesIn, Patients_attendee_2), (Hospital_Model_2, hasRole, Registration_staff_2), (Hospital_Model_2, has Act, Check_weight_and_BP_of_patient_2), (Check_weight_and_BP_of_patient_2, for, Registration_2), (Registration_2, has, Check_weight_and_BP_of_patient_2), (Registration_Staff_2, hasAsParticipant, Registration_2), (Registration_2, participatesIn, Registration_Staff_2)}

G_3={(Hospital_Model_3, hasAct, Allot_Bed_3), (Hospital_Model_3, hasAct, New_Renew_Register_patient_3), Hospital_Model_3, hasAct, Admit_patient_3, (Hospital_Model_3, hasParticipation, Registration_3), (Hospital_Model_3, hasAct, Diagnose_patient_3), (Hospital_Model_3, hasRole, Patients_attendee_3), (Admit_patient_3, for, Registration_3), (Registration_3, has, Admit_patient_3), (Registration_3, has, Allot_Bed_3), (Allot_Bed_3, for, Registration_3), (New_Renew_Register_ patient_3, for, Registration_3),(Registration_3, has, New_Renew_Register_patient_3), (Registration_3, has, Diagnose_patient_3), (Diagnose_patient_3, for, Registration_3), (Patients_attendee_3, hasAsParticipant, Registration_3), (Registration_3, ParticipatesIn, Patients_attendee_3), (Hospital _Model_3, hasRole, Billing_Assistant_3), (Hospital_Model_3, hasEntity, Stethoscope_3), (Hospital_Model_3, hasEntity, BP_Apparatus_3), (Hospital_Model_3, hasRole, Nurse_3), (Hospital_Model_3, hasAct, Check_weight_and_BP_of_patient_3), (Hospital_Model_3, hasRole, Registration_staff_3), (Nurse_3, hasAsParticipant, Registration_3), (Registration_3, ParticipatesIn, Nurse_3), (Registration_3, has, Check_weight_ and_BP_of_patient_3), (Check_weight_and_BP_of_patient_3, for, Registration_3), (Physician_3, plays, Stethoscope_3), (Stethoscope_3, isPlayedBy, Physician_3), (Collect_money_for _consolidated_bill_3, for, Billing_3), (Billing_3, has, Collect_money_for_consolidated_bill_3), (Billing_Assistant_3, has, Billing_3), (Billing_3, ParticipatesIn, Billing_assistant_3), (Prepare_consolidated_bill_3, for, Billing_3), (Billing_3, has, Prepare_consolidated_bill_3)

In step-1, the algorithm compares each triple in G_1 with each triple in G_2 and G_3 to find all possible alignments. Triple 1 of G_1 aligns with A of G_2 and a of G_3, Triple 2 of G_1 aligns with B of G_2 and b of G_3 and so on. Hence the matched triples of G_1, G_2 and G_3 are as follows:
Binding(M) = {(1, A, a), (2 , B , b), (3 , C , c), (4 , D , d), (5 , E , e), (6 , F , f), (7 , G ,g), (8 , H , h), (9 , I , i), (10 , J , j), (11 , K , k), (12 , L , l), (13 , M , m), (14 , N , n), (15 , O , o), (16 , P , p)} = $\{\ell_1, \ell_2, \ell_3, \ell_4, \ell_5, \ell_6, \ell_7, \ell_8, \ell_9, \ell_{10}, \ell_{11}, \ell_{12}, \ell_{13}, \ell_{14}, \ell_{15}, \ell_{16}\}$.

In step-2, hscore is calculated as follows: $hscore(\ell_1) = hscore(\ell_2) = hscore(\ell_3) = 10, hscore(\ell_4) = 18, hscore(\ell_5) = hscore(\ell_6) = 10$, $hscore(\ell_j) = 14$, where $j = 7, 8, .., 16$. Select ℓ_4 and remove it from M and hence the subgraph $SG = \{(4, D, d)\}$.
$Binding(M)$= {ℓ_1, ℓ_2, ℓ_3, ℓ_5, ℓ_6, ℓ_7, ℓ_8, ℓ_9, ℓ_{10}, ℓ_{11}, ℓ_{12}, ℓ_{13}, ℓ_{14}, ℓ_{15}, ℓ_{16}}. The head of ℓ_4 is $(Hospital\_Model\_1/Hospital\_Model\_2/Hospital\_Model\_3)$ is intersecting with $\ell_1, \ell_2, \ell_3, \ell_5, \ell_6$. Remove all from M and hence the subgraph of SG {(4, D, d), (1, A, a), (2, B, b), (3, C, c), (5, E, e), (6, F, f)}. $Binding(M)$ = {ℓ_7, ℓ_8, ℓ_9, ℓ_{10}, ℓ_{11}, ℓ_{12}, ℓ_{13}, ℓ_{14}, ℓ_{15}, ℓ_{16} }. The tail of ℓ_4 is $(Registration\_1/Registration\_2/Registration\_3)$ is intersecting with ℓ_7, ℓ_8, ℓ_9, ℓ_{10}, ℓ_{11}, ℓ_{12}, ℓ_{13}, ℓ_{14}, ℓ_{15}, ℓ_{16}. Hence the subgraph

$$\begin{aligned}SG =\{&(4, D, d), (1, A, a), (2, B, b), (3, C, c),\\ &(5, E, e), (6, F, f), (7, G, g), (8, H, h),\\ &(9, I, i), (10, J, j), (11, K, k), (12, L, l),\\ &(13, M, m), (14, N, n), (15, O, o), (16, P, p)\}.\end{aligned}$$

The result is the maximal subgraph of G_1, G_2 and G_3 which is shown in Figure 4.

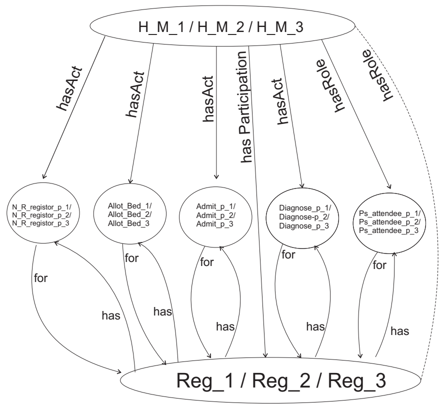
H_M_1 / H_M_2 / H_M_3
hasAct
hasAct
hasAct
has Participation
hasAct
hasRole
hasRole
N_R_registor_p_1/ N_R_registor_p_2/ N_R_registor_p_3
Allot_Bed_1/ Allot_Bed_2/ Allot_Bed_3
Admit_p_1/ Admit_p_2/ Admit_p_3
Diagnose_p_1/ Diagnose-p_2/ Diagnose_p_3
Ps_attendee_p_1/ Ps_attendee_p_2/ Ps_attendee_p_3
for
has
Reg_1 / Reg_2 / Reg_3

Fig 4 Matched Graph

Legend: H_M: Hospital_Model, Reg: Registration, p: patient, N_R: New_Renew

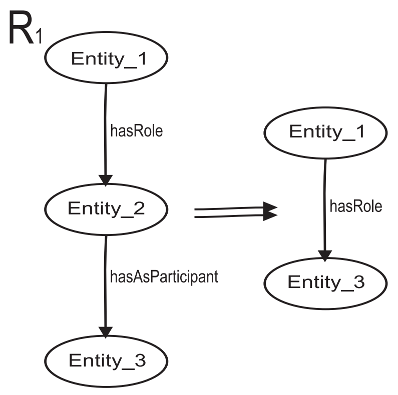

Figure - 5 Transformation

SG is consistent, hence we apply transformations to improve the match. For example transformation R_1 is applied to graphs G_1, G_2 and G_3 and hence a triple is added to graphs G_1, G_2 and G_3. Because of the added triple another match is identified and that is added in SG. The added triple is shown as an edge of dotted line in Figure 4.

B. Fitness of the binding subgraph

In this section we present the fitness of the matched Graph. SG is returned along with a numeric score reflecting the fitness of the match between $G_1, G_2, ..., G_m$. This score is used in situations where one graph is matched with a set of graphs to select the best match. To calculate the fitness score, we apply equation (3) to SG where

$$\begin{aligned} SG = \{&(t_{1j_1}, t_{2j_1}, ..., t_{mj_1}),\\ &(t_{1j_2}, t_{2j_2}, ..., t_{mj_2}), ...,\\ &(t_{1j_k}, t_{2j_k}, ..., t_{mj_k})\} \end{aligned}$$

and the m-tuples $(t_{1j_i}, t_{2j_i}, ..., t_{mj_i})$, for $j = 1, 2, ..., k$ are aligned triples of $G_1, G_2, ..., G_m$ respectively.

$$Fitness(SG) = \frac{\sum_{i=1}^{k} score(t_{1j_i}, t_{2j_i}, ..., t_{mj_i})}{min\{(|G_1| + t_{r1}), (|G_2| + t_{r2}), ..., (|G_m| + t_{rm})\}} \quad (3)$$

where $|G_s|$ is the number of triples in G_s and t_{rs} is the number of additional triples added to G_s, for $s = 1 \cdots m$, by applying transformations. The score for each m-tuple of triples $(t_{1j_i}, t_{2j_i}, ..., t_{mj_i})$ is computed using a function score as given by equation (4).

$$score(t_{1j_i}, t_{2j_i}, ..., t_{mj_i}) = \frac{\frac{1}{x+1} + \frac{1}{y+1} + \frac{1}{z+1}}{3} \quad (4)$$

where, $x = taxdist(head_{1j_i}, head_{2j_i}, ..., head_{mj_i})$, $y = taxdist(rela_{1j_i}, rela_{2j_i}, ..., rela_{mj_i})$, $z = taxdist(tail_{1j_i}, tail_{2j_i}, ..., tail_{mj_i})$ and

$$taxdist(c_1, c_2, ..., c_m) = \sum_{i=1}^{m-1} taxdist(c_i, c_{i+1})$$

the function $taxdist(c_i, c_{i+1})$ is the taxonomic (or semantic) distance [9] between two concepts c_i and c_{i+1}. We calculate the taxonomic distance between two concepts as the minimum number of taxonomic edges that needs to be traversed to reach c_i from c_{i+1}. After transformations have been applied, SG is returned along with a numeric score reflecting the fitness of the match between G_1, G_2,..., G_m. This score is computed based on the number of matched triples over the size of the graphs being matched [11]. The fitness score is also computed using a simple formula as given by equation (5)

$$Fitness(SG) = \frac{No.\ of\ Triples\ of\ SG}{Maximum\ of\ n_1, n_2, ..., n_m} \quad (5)$$

In our example, the fitness of SG is approximately 0.5.

IV. MERGING OF m-CONCEPTUAL GRAPHS

In this section we present an algorithm to merge $m-$ conceptual graphs after determining the matched graph. Let $p_1, p_2, ..., p_m$ be the number of unmatched triples of the graphs G_1, G_2, ... , G_m. Assume that the matched graph SG as G and consider each unmatched triple t_{ji_j} of a graph G_i. Search G to determine the node containing the head of t_{ji_j}. After determining such node, t_{ji} is added as a subnode of the searched graphs node. After adding all the unmatched triples in G, the resultant graph is the merged graph of m Graphs $G_1, G_2, \ldots, G_m$. Returning to our example the merged graph of G_1, G_2,and G_3 is shown in Fig 6.

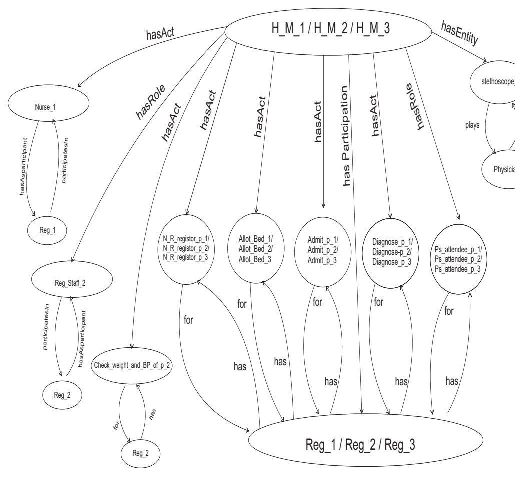

Fig 6 Merged Graph

Legend: H_M: Hospital_Model, Reg: Registration, p: patient, N_R: New_Renew

Algorithm - 2: Outline of the generalized merging algorithm

```
(1) Set SG as G
(2) $FOR$ each unmatched triple $t_{1i_1}$ in $G_1$
    $FOR$ each unmatched triple $t_{2i_2}$ in $G_2$
    ..................................
    $FOR$ each unmatched triple $t_{mi_m}$ in $G_m$
    search $G$ for the head of $t_{ji_j}$, in $n_i$
```

```
      Determine the node where there is a match add $t_{ji_j}$ as
      a subnode for that searched node
      and return the resultant graph as $G$.
(3)   $RETURN\ G$
```

V. Conclusion

Algorithms for matching and merging of $m-$conceptual graphs are developed and the theoretical aspects are discussed in detail. Further these algorithms can be implemented using any ontology management software to support its application to the health care domain.

References

[1] H.Bunke, X.Jiang and A.Kandel, ”On the minimum common supergraph of two graphs”, Computing 65, 1, 2000.
[2] H.Bunke and K.Shearer, ”A graph distance metric based on the maximal common subgraph”, Pattern Recognition Letters, 19,1998.
[3] D.Genest and M.Chein, ”An experiment in document retrieval using conceptual graphs”, ICCS, 1997.
[4] B.T Messmer and H. Bunke, ”A network based approach to exact and inexact graph matching”, Technical report IAM 93-021, Institut fur Informatik, Universitat Bern, 1993.
[5] S.Myacng, ”Conceptual graphs as a framework for text retrieval”, In P. Eklund, T.Nagle, J.Nagle, L.Gerhortz and E.Horwood, editors, current directions in Conceptual Structure Research, 1992.
[6] Peter Z. Yeh, Bruce Porter, Ken Barker, ”Using Transformations to Improve Semantic Matching”, K-CAP’03, October 23-25, 2003, Sanibel Island-florida, USA, Copyright 2003.
[7] A.Sanfeliu and K.Fu, ”A distance measure between attributed relational graphs for pattern recognition”, IEEE Trans.on SMC,13, 1983.
[8] L.Shapiro and R. Haralick, ”Structural descriptions and inexact matching”, IEEE Trans. on PAMI,3, 1981.
[9] J.F.Sowa, ”Conceptual Structures: Information Processing in Mind and Machine”, Addison-Wesley Publishing Company, 1984.
[10] W.Tsai and K.Fu, ”Error-correcting isomorphisms of attributed relational graphs for pattern analysis”, IEEE Trans. on SMC,9, 1979.
[11] P. Yeh, B.Porter and K.Barker, ”Transformation rules for knowledge-based pattern matching”, Technical report UT-AT-TR-03-299, University of Texas at Austin, 2003.
[12] J.Zhong, H.Zhu, J.Li and Y.Yu, ”Conceptual graph matching for semantic search”, ICCS, 2002.