Formal modeling of Gene Ontology annotation predictions based on factor graphs

Flavio Spetale\textsuperscript{1,2}, Javier Murillo\textsuperscript{1,2}, Elizabeth Tapia\textsuperscript{1,2}, Débora Arce\textsuperscript{3}, Sergio Ponce\textsuperscript{3}, Pilar Bulacio\textsuperscript{1,2}

\textsuperscript{1}CIFASIS-Conicet, 27 de Febrero 210 bis, S2000EZP Rosario, Argentina.
\textsuperscript{2}Facultad de Ciencias Exactas, Ingeniería y Agrimensura, Universidad Nacional de Rosario, S2000EZP Riobamba 245 bis, Rosario, Argentina.
\textsuperscript{3}Facultad Regional San Nicolás, Universidad Tecnológica Nacional, Colón 332, San Nicolás, Argentina.

E-mail: spetale@cifasis-conicet.gov.ar

Abstract.
Gene Ontology (GO) is a hierarchical vocabulary for gene product annotation. Its synergy with machine learning classification methods has been widely used for the prediction of protein functions. Current classification methods rely on heuristic solutions to check the consistency with some aspects of the underlying GO structure. In this work we formalize the GO is-a relationship through predicate logic. Moreover, an ontology model based on Forney Factor Graph (FFG) is shown on a general fragment of Cellular Component GO.

1. Introduction
The high-throughput sequencing technologies entail new challenges in data processing. As a result, the use of machine learning algorithms has become relevant in many bioinformatics applications\cite{1,2}. In particular, for Automated Functional protein Prediction (AFP) based on Gene Ontology (GO), ensemble methods consider the ontological structure (DAG, directed acyclic graph) through a hierarchical classification\cite{3,4}. The design of ensemble methods is made in two steps: \textit{i}) In the first one, a set of binary classifiers is built to predict GO-terms (classes), \textit{ii}) In the second one, consistency of GO-DAG relationship is done. Focusing on this last step, different solutions based on heuristics\cite{5,6} have been proposed such as the True Path Rule (TPR) algorithm\cite{5} where the is-a DAG relationship is fulfilled implementing the rule: "If the child term describes the gene product, then all its parent terms must also apply to that gene product".

Heuristic solutions may be a good accuracy-effort trade-off, but a step beyond to attempt logical formalization for checking the DAG restrictions. In this paper we propose that formalization through Forney Factor Graph (FFG) model\cite{7}, since it allows a TPR restriction representation by the logical factorization of functions of several variables associated with GO-terms. The achieved model, called FFG-GO, is able to infer functional predictions of genes by using the sum-product algorithm\cite{8}.

In the next Section a brief background of FFG is presented. Then, in Section 3 TPR restrictions are formalized through predicate logic, in Section 4 FFG-GO model is described. In the last Section, a subgraph of Cellular Component GO is modeled by FFG-GO.
2. Background
An FFG is a model that represents the factorization of a function for several variables. Briefly, the FFG diagram (Fig. 1) has nodes, ordinary edges, and leaf-edges interpreted as follows: each factor $f_i$ (also called function) is represented by a node; each state variable $S_j$ is between two factors and is represented by an ordinary-edge; and each input variable $A_k$ must be involved in just one factor and represented by a leaf-edge. The global function ($f$) is factorized into factors as a product of local functions, where each factor depends on a subset of variables of $f$. For instance, the function $f(A_1, A_2, A_3, A_4, S_1, S_2, S_3)$ can be factorized as:

$$f(A_1, A_2, A_3, A_4, S_1, S_2, S_3) = f_1(A_1, S_1)f_2(A_2, S_1, S_2, S_3)f_3(A_3, S_2)f_4(A_4, S_3)$$

Figure 1: FFG diagram. Boxes are factors, $f_i$ theirs functions; ordinary-edges, $S_j$ theirs state variables; and leaf-edges, $A_k$ theirs input variables.

3. Formalizing GO and TPR restrictions
Aiming to include the GO constraint into FFG model, the TPR restriction is formulated by predicate logic. Next, GO-DAG is denoted by $G = (V, \leq)$, where $V$ is a set of GO terms and $\leq$ is a binary relation on $V$. GO terms are represented by GO nodes ($GO_1, \cdots, GO_m$) where $m$ is the cardinality of $V$. $GO_l \leq GO_z$ denotes that $GO_z$ is a parent of $GO_l$, written in predicate logic as $is_a(GO_l, GO_z)$.

The Gene Ontology Consortium has defined the TPR restriction to guarantee the $is_a$ GO-DAG consistency as follows: “An annotation for a class in the hierarchy is automatically transferred to its ancestors, while unannotated genes for a class cannot be annotated for its descendants”. Note that classes are GO terms and annotation is the process of assigning GO terms to gene products.

Based on [9], we rewrite the TPR by two rules of predicate logic: one related to the parents (Eq.2) and the other related to offspring (Eq.3).

$$r_1 : \forall GO_k \left( is_a(GO_l, GO_k) \land pos(GO_l) \rightarrow pos(GO_k) \right)$$

$$r_2 : \forall GO_k \left( is_a(GO_k, GO_z) \land \neg pos(GO_z) \rightarrow \neg pos(GO_k) \right)$$

where $pos(GO_k)$ means a gene annotation with the $GO_k$ term.

4. FFG-GO model
Regarding the GO behavior modeling under FFG, some GO issues are briefly recalled. The gene ontology is composed of nodes representing gene functions, i.e., the $k$-th node and its function are jointly called $GO:k$. The GO nodes are connected in a DAG structure through edges that
characterize node relationships. Without losing generality, in this work we focus on is-a relation. In Fig. 2(a) a comprehensive GO example that considers multiple offspring-parents is illustrated, while Fig. 2(b) shows its FFG-GO counterpart. GO nodes are matched to FFG model by input variables for root and leaves nodes, and by input state variables for inner nodes. For instance, the leaf GO:5 matches to A5, and the inner node GO:2 matches to A2, S1, and S2. The DAG relationships together with GO restrictions are represented by FFG functions that depend on state and input variables. For instance, r12 and r13 relation match to the function f1.

Figure 2: (a). The GO-DAG example. (b). The FFG-GO counterpart. GO:k node is translated to A_k/S_j variables, GO relationship r are f_i(S_j, A_k).

The strength of FFG model is the design of functions f_i through logical expressions [10] to describe both the native FFG constraints and the GO-DAG restrictions. In the FFG-GO approach we identify three kinds of factors: equality, multiple offspring, and multiple inheritance. Note that the equality is a native FFG factor, but the inclusion of the multiple offspring together with multiple inheritances allows the formal implementation of TPR.

**Equality constraint:** Also called identity function and symbolized with f_= forces all its variables to be equals. It is required in the FFG building [7, 11]. Fig. 3 shows the identity function block, Table 1 describes its truth table, where as expected f_ = 1 when all its variables (A2, S1, S2) are equals.

Table 1: Truth table for equality constraint.

| A2 | S1 | S2 | f_=(A2, S1, S2) |
|----|----|----|----------------|
| 0  | 0  | 0  | 1              |
| 0  | 0  | 1  | 0              |
| 0  | 1  | 0  | 0              |
| 0  | 1  | 1  | 0              |
| 1  | 0  | 0  | 0              |
| 1  | 0  | 1  | 0              |
| 1  | 1  | 0  | 0              |
| 1  | 1  | 1  | 1              |

The f_ = Boolean expression is f_=(A2, S1, S2) = A2 · S1 · S2 + A2 · S1 · S2

Figure 3: Identity function in graphic FFG-GO model
The general form for equality constraint is:

\[ f = \begin{cases} 
1 & \text{pos}(A_k) \land \text{pos}(S_1) \land \ldots \land \text{pos}(S_j) \lor \neg \text{pos}(A_k) \land \neg \text{pos}(S_1) \land \ldots \land \neg \text{pos}(S_j) \quad j = 1, \ldots, n \\
0 & \text{otherwise}
\end{cases} \]

(4)

**Multiple offspring:** Symbolized with \( f \land \), it describes the allowed node states depending on its multiple children states. See Fig. 4 where GO:1 matches to A_1, GO:2 to A_2 and S_1, and GO:3 to A_3 and S_3; Table 2 describes its truth table according to the predicate logic of Eq. 3.

| \( A_1 \) | \( S_1 \) | \( S_3 \) | \( f \land (A_1, S_1, S_3) \) |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Table 2: Truth table for multiple offspring. Inputs: \( A_2 \), \( S_1 \) and \( S_3 \); Output: \( f \land \).

The \( f \land \) Boolean expression is \( f \land (A_1, S_1, S_3) = A_1 + \overline{A_1} \cdot \overline{S_1} \cdot \overline{S_3} \)

The general form for multiple offspring constraint is:

\[ f \land (A_k, S_1, \ldots, S_n) = \begin{cases} 
1 & \text{pos}(A_k) \lor \neg \text{pos}(A_k) \land \neg \text{pos}(S_1) \land \ldots \land \neg \text{pos}(S_j) \quad j = 1, \ldots, n \\
0 & \text{otherwise}
\end{cases} \]

(5)

The \( A_k \) is the FFG input variable that represents the parent GO node, and \( S_1 \) to \( S_n \) are variables that represent its children.

**Multiple inheritance:** Symbolized with \( f \lor \), it describes the allowed node states depending on its multiple parent states. See Fig. 5 where GO:4 matches to A_4 and S_6, GO:2 to A_2 and S_2, and GO:3 to A_3 and S_3; Table 3 describes its truth table according to the predicate logic of Eq. 2.
Table 3: Truth table for multiple inheritance. Inputs: $S_6$, $S_2$ and $S_4$; Output: $f_\lor$.

| $S_6$ | $S_2$ | $S_4$ | $f_\lor(S_6,S_2,S_4)$ |
|-------|-------|-------|---------------------|
| 0     | 0     | 0     | 1                   |
| 0     | 0     | 1     | 1                   |
| 0     | 1     | 0     | 1                   |
| 0     | 1     | 1     | 1                   |
| 1     | 0     | 0     | 0                   |
| 1     | 0     | 1     | 0                   |
| 1     | 1     | 0     | 0                   |
| 1     | 1     | 1     | 1                   |

Figure 5: Multiple inheritance in FFG-GO graphic model.

The $f_\lor$ Boolean expression is $f_\lor(S_6,S_2,S_4) = \overline{S_6} + S_6 \cdot S_2 \cdot S_4$

The general form for multiple inheritance constraint is as follow:

$$f_\lor(A_k, S_1, \ldots, S_n) = \begin{cases} 1 & \neg pos(A_k) \lor pos(A_k) \land pos(S_1) \land \ldots \land pos(S_j) \quad j = 1, \ldots, n \\ 0 & \text{otherwise} \end{cases}$$

(6)

The $S_1$ to $S_n$ are the variables that represents the parent GO nodes, and $A_k$ is the input variable that represent its son.

Summarizing the GO to FFG-GO matching of the multiple offspring-inheritance example, Fig. 2, is shown in Fig. 6.

Figure 6: FFG-GO graphic model of multiple-offspring/inheritance, Fig. 2.

5. Results and discussion

In order to show the strength of the proposed FFG-GO, a model on a rich fragment of GO Cellular Component (CC) is done. This CC subgraph has all the TPR restrictions discussed above, see Fig. 7. For simplicity we have renamed GO nodes, $GO:1$ is the CC root $GO:0005575$, and $GO:2, \ldots, GO:13$ are $GO:0032991$, $GO:0043226$, $GO:0043228$, $GO:0005623$, $GO:0032993$, $GO:0043229$, $GO:0044424$, $GO:0043232$, $GO:0032993$, $GO:0044426$, $GO:0044422$, $GO:0044427$, $GO:0030894$ respectively.

Fig. 8 shows FFG-GO model associated with CC fragment in Fig. 7. The $f$ shows that the GO CC and FFG-GO matching, i.e., GO-terms match directly to $A_k$ input variable. In one hand, the CC root $GO:1$ matches to $A_1$ input variable, and has always associated with the multiple offspring function $f_\land$. On the other hand, leaves ($GO:9,GO:13$) match to $A_9, A_{13}$ input variables, and have always associated with the multiple inheritance function $f_\lor$. We should note
that TPR restrictions are associated just with FFG-GO functions, hence, a restriction change requires a function modification but not a FFG-GO model rebuilding as happens with other approaches.

The proposed FFG-GO model allows the formalization of is-a DAG relationship of GO ontology. Without loss of generality, the inclusion of new DAG relationships or restrictions
requires just a redesign of functions $f$ without changing the core of FFG-GO model. Likewise, the insertion of new GO-terms is simple and easy to interpret graphically. In order to complete the GO-term annotation prediction for each sample (protein), a dynamic inference process must takes place on the FFG-GO model. This inference process is carried out by the sum-product algorithm, but its description is out of the scope of this work, see [8] for details.

6. Conclusion
In ensemble classification methods, GO-term predictions computed by base binary classifiers are leveraged by checking the consistency of predefined GO relationships. Both formal leveraging strategies, with main focus on annotation precision, and heuristic alternatives, with main focus on scalability issues, have been described in literature.

Focusing on formal strategies, the formalization of TPR restrictions by predicate logic. The further work is to embody this formalization in a hierarchical classification method based on graphical models.

Along this paper we have focused our attention on is-a relationship of GO. However, this approach may be extended to another types of transitive relationships, such as part of, has part.

Acknowledgments
The authors were supported by projects PID INI3600; PICT 0253; PICT 2513, Argentina.

References
[1] Altschul S F, Gish W, Miller W, Myers E W and Lipman D J 1990 Journal of Molecular Biology 215 403–410 ISSN 0022-2836
[2] Edgar R C 2004 Nucleic Acids Research 32 1792–1797
[3] Cheng L, Lin H, Hu Y, Wang J and Yang Z 2014 PLoS ONE 9 e107187
[4] Schietgat L, Vens C, Struyf J, Blockeel H, Kocev D and Dzeroski S 2010 BMC Bioinformatics 11 2
[5] Valentini G 2011 Computational Biology and Bioinformatics, IEEE/ACM Transactions on 8
[6] Stojanova D, Ceci M, Malerba D and Dzeroski S 2013 BMC Bioinformatics 14 285 ISSN 1471-2105
[7] Forney GD J 2001 Information Theory, IEEE Transactions on 47 520–548 ISSN 0018-9448
[8] Kschischang F R, Frey B J and Loeliger H A 2001 IEEE Trans. Inf. Theor. 47 498–519
[9] Burger A, Davidson D and Baldock R A 2004 Bioinformatics 20 259–267
[10] Morcos F, Sikora M, Alber M, Kaiser D and Izaguirre J 2010 Information Theory, IEEE Transactions on 56 742–755
[11] Loeliger H 2004 IEEE Signal Processing Magazine 21 28–41 ISSN 1053-5888