Token Evolution in an Ill-posed Constraint Petri-net

Hideaki Suzuki
Center for Information and Neural Networks
(CiNet)
National Institute of Information and
Communications Technology (NICT)
588-2, Iwaoka, Iwaokaka-cho, Nishi-ku, Kobe,
651-2492 Japan
hsuzuki@nict.go.jp

Yoh Iwasa
Department of Biology
Faculty of Science
Kyushu University
6-10-1, Hakozaki, Higashi-ku, Fukuoka,
812-8581 Japan
yohiwasa@kyudai.jp

ABSTRACT

Unification constraints of predicate logic or string rewriting rules of context free grammar can be represented by a Petri-net with the AND/OR tree structure. In this network, deductive inference or parsing can be formulated as a problem to find a solution subtree within which variables have consistent substitution. The paper focuses on a token-based heuristic method to solve such an ill-posed problem, and presents three prerequisite conditions to be satisfied by tokens which propagate and evolve in the network. Incorporating these requirements would enable us to design highly distributed algorithms for deduction or parsing.

Categories and Subject Descriptors

I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving—*Logic programming*

Keywords

Petri-net, AND/OR-tree, Deductive inference, Token, Ill-posed problem

1. INTRODUCTION

For a long time, researchers in the computer science have tried to implement intelligence in a computer in two different ways: a way to manipulate symbols within the framework of Artificial Intelligence [15] and a softer and more resilient way that borrows biological mechanisms and emulates such systems as neural networks [6, 2], biological evolution [5], and so on) (this latter approach is sometimes called Computational Intelligence nowadays). Above all, as exemplified by artificial neural networks [6], network representation has been regarded as a metaphor of brain, and has been most often taken as a representation scheme within the latter approach.

This situation, however, is gradually changing after we began to take a third, intermediate approach between symbols and emulation, a way to manipulate symbols in artificial networks. One of the most representative models in this approach would be Bayesian networks which explicitly cope with probability with prepared transition tables [2, 4, 9]. Besides that, Suzuki *et al.* recently extended a historic approach named High-level Petri-net [8, 16, 12, 13, 10, 7] and proposed a novel network model named Knowledge Transitive Networks – data-flow-based Knowledge Transitive Network (dKTN) [17, 18, 21] and Petri-net-based Knowledge Transitive Network (pKTN) [22, 24, 25]. The distinctive features of the KTNs are summarized as follows:

- Knowledge is purely represented by the topological structure of a network. This is in contrast with the High-level Petri-net which represents predicate logic by the aid of arc labels.
- They are a kind of direct representation of first-order predicate logic including variables and function symbols. This is in contrast with the Bayesian networks which basically propositionalizes first-order logic (i.e., eliminates variables) for probabilistic reasoning.
- Deductive inference is propelled by a method named ELISE (ELiminating Inconsistency by SElection) [19, 20, 18, 21] which uses parallel propagation and evolution of tokens in a network. This could enable us to represent a kind of 'ambiguity' in inference.

Although the KTN was originally introduced to visualize a logic program, it (particularly the pKTN) is intimately related with constraint programming (CP) which tries to find consistent substitution for variables under a set of prepared constraints [26, 1]. The CP is sometimes represented using a constraint network whose nodes represent variables and whose edges represent constraints [11, 3]. The pKTN's places stand for variables and transitions stand for unification constraints, so in this sense, the pKTN can be regarded as a kind of constraint networks; and yet, these two have striking difference in the following point.

When we solve constraint satisfaction problems by the CP, their solutions are usually required to satisfy all the constraints (all the edges in constraint networks). Whereas in the pKTN, a solution (variables' substitution) is required to satisfy only a part of the pKTN's expansion tree which can grow semi-infinitely. This makes a problem 'ill-posed', and to solve it, we have to evaluate a solution under two criteria, appropriate choice of a subnet in the network and variables' consistency within the subnet. The present paper focuses on this type of problems in a constraint network, and studies a way to deal with them. Specifically, the paper takes ELISE that solves deduction in the pKTN and theoretically analyzes its token dynamics.

BIICT 2014, December 01-03, Boston, United States

DOI 10.4108/icst.bict.2014.257936

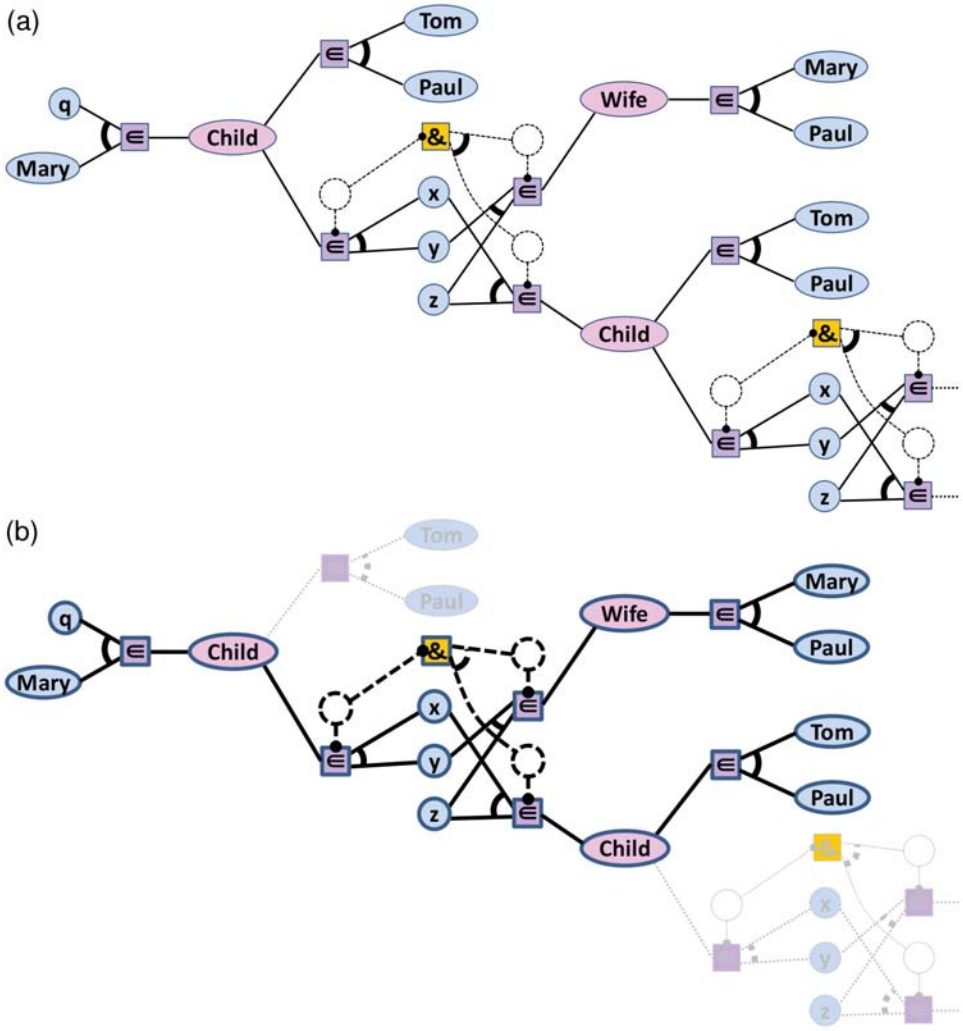

Figure 1: (a) Expanded pKTN for logic program (1a)∼(1d), and (b) its solution subtree.

In the following, we first present two typical examples of constraint Petri-nets with ill-posedness (pKTN and AND/OR graph representing generative grammar) in Section 2. After briefly describing the basic ways of ELISE in Section 3, we consider how to propagate and evolve tokens in ELISE and present three necessary conditions to be satisfied by the tokens in Section 4. Section 5 gives concluding remarks.

2. CONSTRAINT PETRI-NETS

Let us consider a logic program for family relationship

$$\text{[Fact]} \quad Wife(Mary, Paul). \tag{1a}$$

$$\text{[Fact]} \quad Child(Tom, Paul). \tag{1b}$$

$$\text{[Rule]} \quad Child(x, y) \leftarrow Wife(y, z), Child(x, z). \tag{1c}$$

$$\text{[Goal]} \quad \leftarrow Child(q, Mary). \tag{1d}$$

Here, $Wife$ and $Child$ are predicates, $Paul$, $Mary$, and Tom are constants, x, y, and z are variables, and q is a specific query variable. Since this program has predicate $Child$ in both of the head and body of clause (1c), the expanded pKTN for this program is a semi-infinite AND/OR tree [22, 24, 25]. Figure 1(a) shows the first three tiers of the tree, and Fig. 1(b) shows its solution subtree. Here and in Fig. 2(b), the solution tree ('true branches') is drawn in thick bold lines, and the remaining parts ('false branches') are drawn in light dotted lines. In all the figures, transitions are expressed as squares, and places are expressed as circles/ellipses. In Fig. 1, the place circles have their constants', variables', or predicates' names; however, only constants' names have significant meaning for deduction. When firing, a transition with symbol '$\in$' splits/combines token terms (splits a term vector into elements or combines terms into a vector), and that with symbol '&' calculates 'Logical AND'. According to [21], it is proved that the existence of a consistent solution tree in an expanded pKTN is equivalent to the truthfulness of the whole expansion tree under some unification, namely

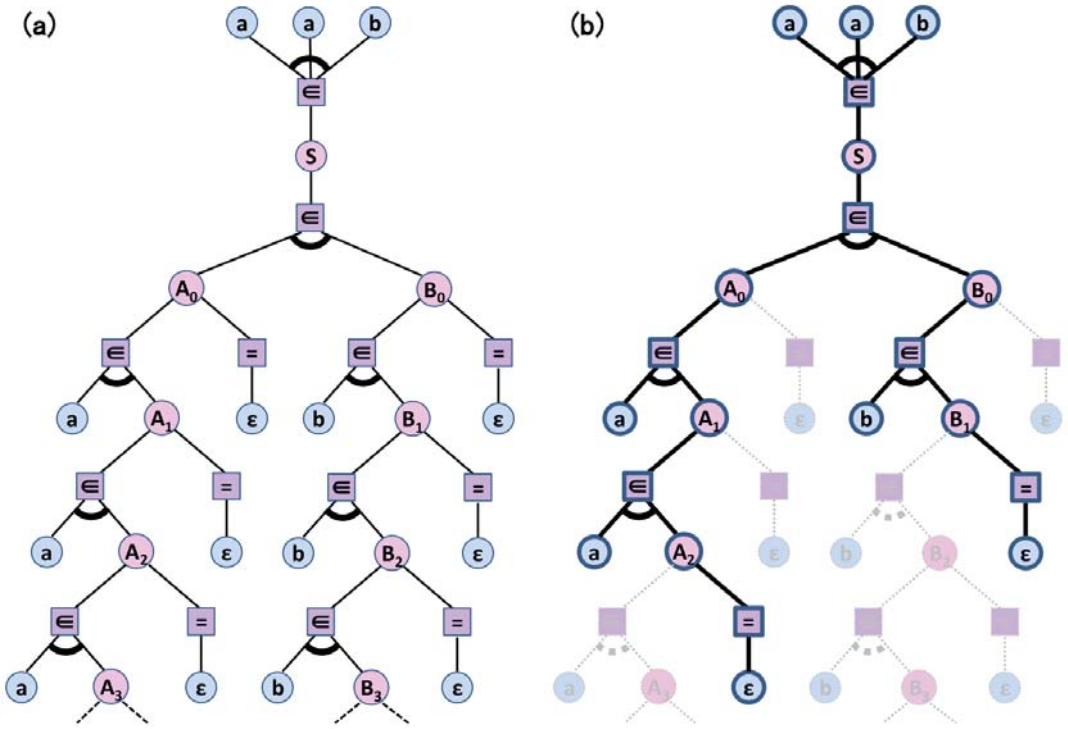

Figure 2: (a) AND/OR graph for context free grammar (2) with input sequence aab, and (b) its solution subtree.

the existence of a solution for the deductive inference (soundness and completeness).

Another constraint Petri-net we argue here is one obtained from generative grammar in the formal language theory. Let us consider the following context free grammar that generates language $\{a^m b^n | m, n \geq 0\}$:

$$\left.\begin{array}{lcl} S & \rightarrow & AB \\ A & \rightarrow & aA \\ A & \rightarrow & \epsilon \\ B & \rightarrow & bB \\ B & \rightarrow & \epsilon \end{array}\right\}. \qquad (2)$$

Here, S is the starting symbol, A, B are non-terminal symbols, a, b are terminal symbols, and ϵ is the empty symbol. Since non-terminal symbols A and B appear on both sides of the rewriting rules, an AND/OR tree that graphically represents derivation by this grammar is also semi-infinite. Figure 2(a) shows its first three tiers with a constraint part by the input sequence aab. Here, a transition has a function for 'concatenate' ($\in$) or 'equal' ($=$) depending on the number of operands concatenated. With this AND/OR tree, syntax analysis is formulated as the search for a solution subtree within which non-terminal symbols (variables) have consistent symbol sequences with the constraints [23].

In these examples, deduction or parsing can be regarded as an ill-posed problem which seeks an appropriate subtree out of the semi-infinite tree wherein variables have consistent substitution. In the next section, we argue a method to solve this problem by a token-based algorithm.

3. ELISE

ELISE (ELiminating Inconsistency by SElection) is a method to solve a system of simultaneous constraint equations by using a data-flow network or Petri-net that represents constraints.

As is well known, when we use a Petri-net for parallel computation, the calculation is propelled by the local operation (firing) of a transition that changes the numerical value of a token [14]. ELISE extends this value (denoted by x) to symbol, term (sequences of symbols), or term list, and at the same time, tags a token with a real number named 'reliability' (denoted by r), that is a measure for inter-token consistency. While tokens propagate in the network, evolutionary operations take place in pools prepared at some places of the network, which makes tokens with larger rs survive and the xs converge to ones consistent with the constraint conditions.

So far, ELISE has been applied to data-flow networks representing a system of numerical equations [19, 20] and symbolic equations [18, 21], demonstrating its ability to solve the equations asymptotically/approximately. However, the constraint networks of these experiments had no 'redundancy', and tokens had to be consistent assuming that all the constraints in the network are valid. (In the case of numerical equations, some inconsistency was often included, but in this case as well, ELISE was required to solve them approximately taking all the constraints into consideration.) When we solve the ill-posed problem, on the other hand, the situation is very different. Tokens have to find consistent substitution of vari-

ables in a subnet while choosing the subnet itself. If the chosen subnet is too large, the variables cannot be consistent, whereas if the chosen subnet is too small, tokens will converge to trivial values with no meaning.

In the next section, we present some prerequisite conditions to be satisfied by evolutionary operations of tokens, which will help us design token-based algorithms like ELISE for the ill-posed problem.

4. PREREQUISITE CONDITIONS FOR TOKEN EVOLUTION

(1) Information about small r caused by inconsistency should be transmitted through the entire network. We consider a token population distributed through a constraint Petri-net. At a place of this network, let us assume that a token has come to have a small r on account of its inconsistency with neighboring tokens. If we evolve tokens constantly and locally at places in the network, this token will soon die out by selection, but this causes disadvantage from the perspective of evolution of the whole population. See the expanded pKTN in Fig.1. At first, the backward (leaf to root) token propagation in this Petri-net makes tokens arrive at an OR ($Child$) place from both true and false branches. Out of these, a token from a false branch later causes inconsistency at a distant z place, for example, but this information is important to a token that goes to the q place as well. If this information on inconsistency at the z place is not transmitted to the token at the q place, the token from the false branch might survive at the q place for a long time, which can hinder the population from converging to the correct answer. In order to stop this, information on inconsistency of a token at a place should be swiftly transmitted to all of the tokens consistent with the token and had in common.

(2) Inconsistency at a place inside the solution subtree and that surrounding the solution subtree should be discriminated. Let us assume that the former requirement was met and we have actualized a scheme that weeds out all the tokens consistent with an inconsistent token at a place. This scheme, however, causes another problem in the network. Again, we take a solution subtree (thick bold tree) in Fig.1(b) as an example. When this subtree is successfully extracted and all the pools inside of the subtree are filled with correct (consistent) tokens, the two $Child$ places in the subtree will be dominated by $(Tom, Mary)$ tokens (on the left side) or $(Tom, Paul)$ tokens (on the right side). If we prohibit a transition from creating a value inconsistent with a constant place, a $(Tom, Mary)$ token at the left $Child$ place does not cause a problem, i.e., does not cause the firing of the upper right transition in the false branch; whereas with a $(Tom, Paul)$ token at the right $Child$ place, this is not the case. The right $Child$ place's lower right transition in the false branch has x, y variable places on the right-hand side, and can always fire and create a Tom token at the x and a $Paul$ token at the y place. These tokens can be inconsistent with tokens from the false branch; hence if this was transmitted to tokens in the solution subtree, they might be also exterminated. The tokens in the solution subtree cannot stably exist without some scheme that discriminates between inconsistency at an inside place and that at a peripheral place.

(3) AND/OR structure of the constraints should be considered. On top of the former conditions, here we require that the control mechanisms of token propagation should incorporate the overall logical (AND/OR) structure of the network in order for a solution subtree to be properly extracted and kept stably. In such a model as ELISE wherein token propagation is triggered by the transition firing, this requirement would be actualized by a scheme that regulates the transition firing (by adjusting the firing probability, etc.) or makes each token choose appropriate arcs during propagation.

5. CONCLUSION

Using the constraint Petri-nets that represent such declarative knowledge as predicate logic or generative grammar, deductive inference or parsing on context free grammar can be formulated as an ill-posed problem that requires extraction of a subnet from the network and consistency between variables' substitution within the subnet. A token-based method to solve this type of problems was taken in this paper, and we presented three prerequisite conditions to be satisfied by the token evolution and propagation. Future research in this line includes the design of a concrete method that meets these requirements and its evaluation by some numerical experiments.

6. REFERENCES

[1] Barták, R.: Constraint Programming: In Pursuit of the Holy Grail. In: Proceedings of the Week of Doctoral Students (WDS99), Part IV. MatFyzPress, Prague (1999) 555-564

[2] Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag (2006)

[3] Dechter, R.: From local to global consistency. Proceedings of the 8th CSCSI (Canadian AI Conference), Ottawa, Canada (1990) pp. 231-237 (best paper award) Artificial Intelligence **55**(1) (1992) 87-107

[4] De Raedt, L.: Logical and Relational Learning (Cognitive Technologies). Springer (2008)

[5] Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing (Natural Computing Series). Springer (2003)

[6] Haykin, S.: Neural networks and learning machines. Prentice-Hall, Inc. (2009)

[7] Jeffrey, J., Lobo, J., Murata, T.: A high-level Petri net for goal-directed semantics of Horn clause logic. IEEE Transactions on Knowledge and Data Engineering **8**(2) (1996) 241-259 DOI: 10.1109/69.494164

[8] Jensen, K.: Coloured Petri Nets and the Invariant Method. Theoretical Computer Science **14** (1981) 317-336 North-Holland Publishing Company.

[9] Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques (Adaptive Computation and Machine Learning series). The MIT Press (2009)

[10] Lin, C., Chaudhury, A., Whinston, A.B., Marinescu, D.C.: Logical Inference of Horn Clauses in Petri Net Models. IEEE Transactions on Knowledge and Data Engineering **5**(3) (1993) 416-425

[11] Montanary, U.: Networks of constraints fundamental properties and applications to picture processing. Information Sciences 7 (1974) 95-132

[12] Murata, T., Zhang, D.: A predicate-transition net model for parallel interpretation of logic programs. IEEE Transactions on Software Engineering *14*(4) (1988) 481-497

[13] Peterka, G., Murata, T.: Proof procedure and answer extraction in Petri net model of logic programs. IEEE Transactions on Software Engineering **15**(2) (1989) 209-217 DOI: 10.1109/32.21746

[14] Peterson, J.L.: Petri Net Theory and the Modeling of Systems. Prentice Hall (1981)

[15] Poole, D.L., Mackworth, A.K.: Artificial Intelligence. Cambridge University Press (2013)

[16] Reisig, W.: Petri nets: an introduction. Springer-Verlag, New York, Inc. New York, NY, USA (1985)

[17] Suzuki, H., Yoshida, M., Sawai, H.: A proposal of data-flow network for deductive inference. In: The Special Interest Group Notes of the

Japanese Society for Artificial Intelligence: Fundamental Problems of Artificial Intelligence. SIG-FPAI-B102 (2011) 1-7 (Japanese)

[18] Suzuki, H., Yoshida, M., Sawai, H.: A data-flow network that represents first-order logic for inference. In: Kuo, Y.H., Tseng, V.S.M., Kao, H.Y., Hong, T.P., Horng, M.F. (eds.): The 2012 Conference on Technologies and Applications of Artificial Intelligence TAAI, Proceedings (2012) 211-218 DOI: 10.1109/TAAI.2012.44

[19] Suzuki, H.: Proposal of a data-flow network that solves simultaneous equations by back-calculation and evolution. Proceedings of the 57th Annual Conference of the Institute of Systems, Control and Information Engineerings (SCI'13). ISCIE, Kobe (2013) 317-4 (Japanese)

[20] Suzuki, H., Iwasa, Y.: A network-based evolutionary method to solve inconsistent simultaneous equations approximately. AIP Conf. Proc. **1558** (11th International Conference of Numerical Analysis and Applied Mathematics - ICNAAM) (2013) 2486-2491 doi: 10.1063/1.4826045

[21] Suzuki, H., Yoshida, M., Sawai, H.: A network representation of first-order logic that uses token evolution for inference. Journal of Information Science and Engineering (JISE) **30**(3) (2014) 669-686

[22] Suzuki, H., Yoshida, M.: Logical representation and deduction by a Petri-net-based KTN with predicates as variable nodes In: Proceedings of the 41st SICE Symposium on Intelligent Systems, The Society of Instrument and Control Engineers (2014) B22-3 (Japanese)

[23] Suzuki, H.: Proposal of a syntax analyzer that allows ambiguity in input words. In: Proceedings of the 20th Annual Meeting of The Association for Natural Language Processing (2014) 428-431 (Japanese)

[24] Suzuki, H., Yoshida, M.: Direct graphical representation of first-order logic for inference. In: Lambert, M.J. (ed.): Logic Programming: Theory, Practices and Challenges. Chap.4. Nova Science Publishers, Inc. (2014) 117-142 ISBN: 978-1-63117-853-5

[25] Suzuki, H.: A method to solve an ill-posed problem in networks representing constraints In: Proceedings of the fifth Meeting of the Special Interest Group, Computational Intelligence. The Society of Instrument and Control Engineers (SICE) PG0008/14/0000-0037 (2014) (Japanese)

[26] Van Hentenryck, P.: Constraint Satisfaction in Logic Programming. MIT Press, (1989)