Formal Models of the Structural Errors in the Knowledge Bases of Intellectual Decision Making Systems

Olga Dolinina(✉) and Natalya Suchkova

Yury Gagarin State Technical University of Saratov, Saratov, Russia
odolinina09@gmail.com, rinoa_27@mail.ru

Abstract. The paper provides classification of structural errors in the rule-based knowledge bases of intellectual decision making systems. For detecting of the structural errors it is proposed to use AND/OR graph representing a knowledge base of rule-based system. There are described formal models of the 3 types of structural errors identified: redundancy errors, incompleteness errors and inconsistency. Redundancy errors are described with duplicates, redundant inference chains, insignificant inference chains, incorrect inference chains and cycles. Incompleteness error example is isolated vertices. Inconsistency in knowledge bases is represented by conflicting inference chains. For each structural error the paper provides formalization in terms of the graph model and the way of correction.

Keywords: Debugging of the intellectual decision making systems · Rule-based systems · Static analysis · Verification · Structural errors · AND/OR graph

1 Introduction

Intellectual decision making systems (IDMS) are used in wide range of areas: industry, medicine, research activities, education, classification tasks including critical areas. It demands reliability of making decisions by these systems what depends on all components of the IDMS. Methods and algorithms of improvement of the reliability of the software and hardware components are well developed but providing of the quality of the knowledge base (KB) is still a problem which has not been solved yet. Methods of knowledge bases (KB) debugging are still not formalized. Algorithms of the traditional software debugging cannot be used for the knowledge bases and most of the developers use expert approach for the debugging of the knowledge bases.

Quality of KB is a multi-criteria problem, there are different approaches for the checking of the correctness and completeness including methods of detecting of various types of errors [1–7], but debugging of the KB is still considered to be the most complicated stage of the IDMS development. There are errors in the knowledge base which are connected with the inconsistency of the knowledge area – for example, errors of forgetting-about-the-exception type [2] and which can be detected by the testing only. At the same time, so called, structural errors can be detected and deleted from the

R. Silhavy et al. (eds.), *Artificial Intelligence Trends in Intelligent Systems*,
Advances in Intelligent Systems and Computing 573, DOI 10.1007/978-3-319-57261-1_16

KB on the stage of the static debugging (the formal checking of the KB) which does not demand the running of the intellectual system.

The absence of the structural errors does not guarantee the absence of the errors connected with the inconsistency of the knowledge area but it increases the effectiveness of the decision making due to the reducing of the solution time, so the first necessary step of the debugging of the KB is the formal checking (static analysis).

In the papers [3–7] several structural errors in the KB and algorithms for their detecting are described but no errors formalization has been made and the current paper accumulates full information on structural errors and provides formal models of the errors which can be used as a basis for automatic verification of knowledge bases structure.

2 Formal Models of Structural Errors of the Rule-Based Knowledge Base

Rules are widely used for representing of the knowledge bases of intellectual decision making systems. Rule-based knowledge base is set as:

$$P = (F, R, G, C, I), \tag{1}$$

where F is a finite set of the facts in the concrete field about the problem.

R – a set of rules where

$$r_m : IF\, f_i\, and\, f_j \ldots and\, f_n\, then\, f_k, \tag{2}$$

G – set of goals or the IDMS terminal facts; C is the set of the permitted combination of the facts; I – the interpreter of the rules, realizes the goal solution.

Let S is the set of input facts, i.e. facts specified by the user in the input of the intellectual system. $S \subset F$.

The logic of the knowledge base can be presented by the AND/OR graph. For example, let's build AND/OR graph for a set of the following rules:

$r_1 : if\, s_1\, and\, s_2,\, then\, f_1;$
$r_2 : if\, s_2\, and\, s_3,\, then\, f_2;$
$r_3 : if\, s_3\, and\, s_4\, and\, s_5,\, then\, f_3;$
$r_4 : if\, f_1,\, then\, g_1;$
$r_5 : if\, f_2\, and\, f_3,\, then\, g_2.$

Figure 1 provides an example of AND/OR graph, where:

$s_1, s_2, s_3, s_4, s_5 \in S; f_1, f_2, f_3 \in F; r_1, r_2, r_3, r_4, r_5 \in R; g_1, g_2 \in G.$

Rule r_i as

$$r_{i:}\ \text{if}\, f_{r_i1}\ \text{and}\, f_{r_i2} \ldots f_{r_in},\ \text{then}\, f_{r_im}$$

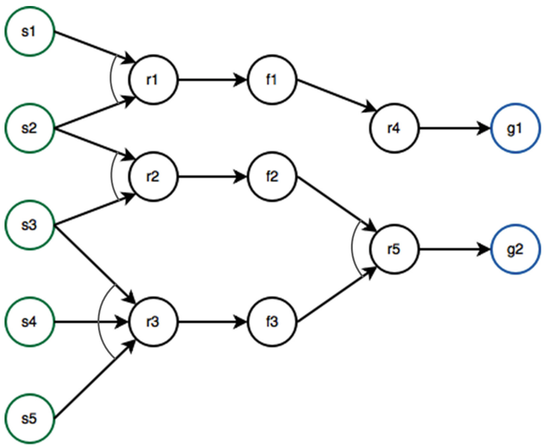

Fig. 1. AND/OR graph example

can be represented as a pair $\mathrm{r_i} = (D_{r_i}; Q_{r_i})$, where $D_{r_i} = \{f_{r_i}1, f_{r_i}2, ..f_{r_i}n\}$ and $Q_{r_i} = \{f_{r_i}m\}$. Q_{r_i} has a single element, henceforward defined as q_{r_i}.

Let L is a set of inference chains.

Definition 1. An inference chain l_i – is a sequence of rules $(r_{l_i}1, r_{l_i}2, \ldots, r_{l_i}n)$, if $\forall r_{l_i}k, r_{l_i(k+1)}, q_{r_{l_ik}} \in D_{r_{l_i(k+1)}}$, where $k = 2, \ldots, (n-1)$.

Then, the graph on Fig. 1 has $L = \{l_1, l_2, l_3, l_4, l_5, l_6, l_7, l_8\}$, where

$$l_1 = (r_1);\ l_2 = (r_2); l_3 = (r_3); l_4 = (r_4); l_5 = (r_5); l_6 = (r_{1,}r_4);$$
$$l_7 = (r_2, r_5); l_8 = (r_3, r_5).$$

Definition 2. The start of inference chain l_i as $(r_{l_i}1, r_{l_i}2, \ldots, r_{l_i}n)$, is a set of the facts in condition of the first rule of the chain, $D_{l_i} = D_{r_{l_i}1}$.

Definition 3. The end of inference chain l_i as $(r_{l_i}1, r_{l_i}2, \ldots, r_{l_i}n)$ is a result of the last rule of the chain, $Q_{l_i} = Q_{r_{l_i}n}$.

Definition 4. Structural error in the rule-based system is an error which can be detected during AND/OR graph analysis. Knowledge base which doesn't have any structural errors is considered as a statically correct one.

The classification of structural errors is provided in Fig. 2.

2.1 Redundancy Errors

2.1.1 Duplicates

Definition 5. Rules r_i and r_j are considered as duplicates, if $D_{r_i} \cap D_{r_j} \neq \emptyset$ and $q_{r_i} = q_{r_j}$.

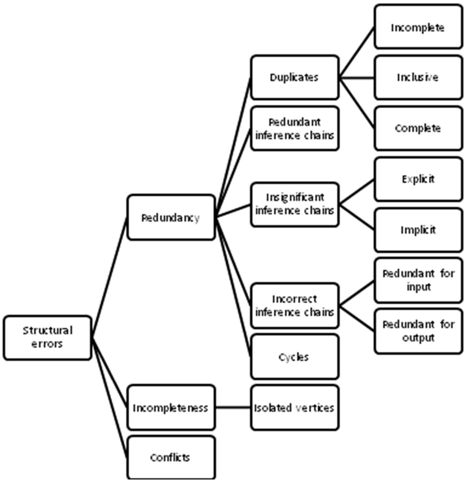

Fig. 2. Structural errors classification

There are three types of duplicates:

- inclusive duplicates;
- complete duplicates;
- incomplete duplicates.

Definition 6. Rules r_i and r_j are considered as inclusive duplicates, if $D_{r_i} \subset D_{r_j}$. $D_{r_i} \neq D_{r_j}$ and $q_{r_i} = q_{r_j}$, r_i is defined as included one in this case.

Rules r_1 and r_2 in the Fig. 3. are inclusive duplicates:

$$
\begin{aligned}
&r_1 : \; if\, f_1 \; and\, f_2, \; then\, f_4; \\
&r_2 : \; if\, f_1 \; and\, f_2 \; and \; f_3, \; then\, f_4; \\
&D_{r_1} = \{f_1, f_2\}; \quad D_{r_2} = \{f_1, f_2, f_3\}; \quad D_{r_1} \,\&\, D_{r_2} \\
&q_{r_1} = f_4; \quad q_{r_2} = f_4;
\end{aligned}
$$

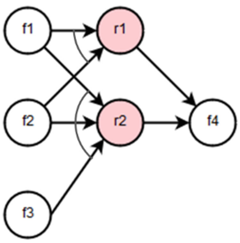

Fig. 3. Inclusive duplicates

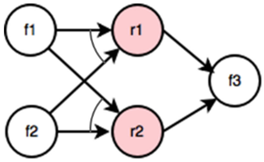

Fig. 4. Complete duplicates

The solution to correct inclusive duplicates error is to remove all duplicating rules except included one.

Definition 7. Rules r_i and r_j are considered as complete duplicates, if $D_{r_i} = D_{r_j}$ and $q_{r_i} = q_{r_j}$.

Rules r_1 and r_2 in the Fig. 4. are complete duplicates:

r_1 : *if* f_1 *and* f_2, *then* f_3;
r_2 : *if* f_1 *and* f_2, *then* f_3;

The solution to correct complete duplicates error is to remove all duplicating rules except one.

Definition 8. Rules r_i and r_j are considered as incomplete duplicates, if $D_{r_i} \cap D_{r_j} \neq \emptyset$, $q_{r_i} = q_{r_j}$, $D_{r_i} \backslash D_{r_j} \neq \emptyset$ and $D_{r_j} \backslash D_{r_i} \neq \emptyset$.

The Fig. 5. provides an example of incomplete duplicates:

r_1 : *if* f_1 *and* f_2, *then* f_4;
r_2 : *if* f_2 *and* f_3, *then* f_4;
$D_{l_1} \cap D_{l_2} = f_2$;

There is no single solution for correcting incomplete duplicates errors, so the concrete decision on the way of the improvement of the KB and correctness the error should be made by the expert in each particular case.

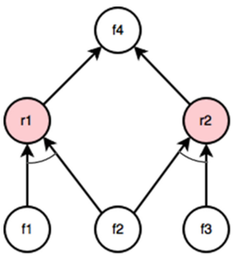

Fig. 5. Incomplete duplicates

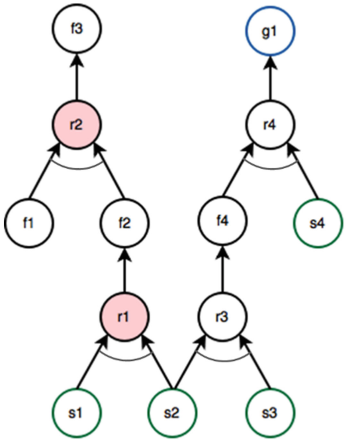

Fig. 6. Redundant inference chain

2.1.2 Redundant Inference Chains

Definition 9. An inference chain l_i is as considered redundant, if $q_{l_i} \notin G$ and $\neg \nexists l_j$, where $q_{l_i} \in D_{l_j}$ and $q_{l_j} \in G$.

The Fig. 6 provides an example of redundant inference chain $-l_1 = (r_1, r_2)$, where:

r_1 : *if* s_1 *and* s_2, *then* f_2;
r_2 : *if* f_1 *and* f_2, *then* f_3;
$q_{l_1} = f_3; f_3 \notin G$;

Redundant inference chains can be removed from a knowledge base.

2.1.3 Insignificant Inference Chains

Definition 10. An inference chain l_i as $(r_{l_i1}, r_{l_i2}, \ldots, r_{l_in})$ is considered as insignificant one, if $\forall r_{l_ij}, \left|D_{r_{l_ij}}\right| = 1$, where $j = 1, \ldots, n$.

$l_1 = (r_1)$ in the Fig. 7. is an insignificant inference chain:

$D_{r_1} = \{f_1\}; \quad |D_{r_1}| = 1;$

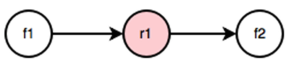

Fig. 7. Insignificant inference chain

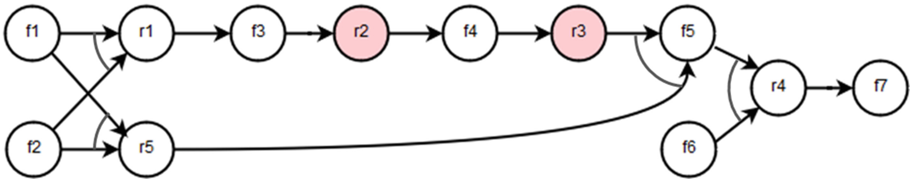

Fig. 8. Explicit insignificant inference chain

There are two types of insignificant inference chains:

- explicit chains;
- implicit chains.

Definition 11. An insignificant inference chain l_i is as considered explicit one, if $\exists r_j$, where $r_j = (D_{l_i}; Q_{l_i})$.

The Fig. 8 provides an example of explicit insignificant inference chain $l_1 = (r_2, r_3)$, where:

$r_2 : \textit{if}\, f_3, \textit{then}\, f_4;$
$r_3 : \textit{if}\, f_4, \textit{then}\, f_5;$
$D_{l_1} = \{f_3\}; q_{l_1} = \{f_5\};$
And the rule $r_5 = \{D_{l_1}; q_{l_1}\}$ *exists.*

The solution is to remove explicit insignificant inference chain.

Definition 12. An insignificant inference chain l_i is as considered implicit one, if $\neg\exists r_j$ where $r_j = (D_{l_i}; Q_{l_i})$.

The Fig. 9. provides an example of implicit insignificant inference chain $l_1 = (r_2, r_3)$, where:

$r_2 : \textit{if}\, f_3, \textit{then}\, f_4;$
$r_3 : \textit{if}\, f_4, \textit{then}\, f_5;$
$D_{l_1} = \{f_3\}; q_{l_1} = \{f_5\}; r_5 = \{D_{l_1}; q_{l_1}\}$
And the rule $r_k = \{D_{l_1}; q_{l_1}\}$ *doesn't exist.*

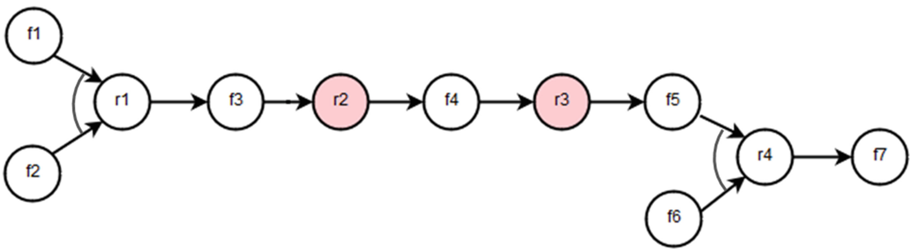

Fig. 9. Implicit insignificant inference chain

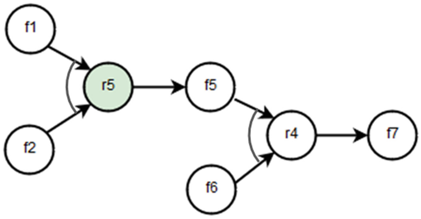

Fig. 10. Transforming to the preceding rule

An implicit insignificant inference chain is not a critical one for the knowledge base, but it allows optimization by transforming to the preceding or succeeding rule.

Definition 13. Transforming to the preceding rule r_n of the insignificant inference chain l_i where $q_{r_n} \in D_{l_i}$, is a creation of a rule r_m, where $r_m = (D_{r_n}; Q_{l_i})$, and deletion of l_i and r_n from the knowledge base.

The transformation to the preceding rule of the insignificant inference chain in the Fig. 9 is shown in the Fig. 10. A new rule r_5 has been added:

r_5 : *if* f_1 *and* f_2, *then* f_5;

Definition 14. Transforming to the succeeding rule r_n of the insignificant inference chain l_i where $q_{l_i} \in D_{r_n}$, is a creation of the rule r_m, where $r_m = ((D_{r_n} - q_{l_i}) \cup D_{l_i};$ $Q_{r_n})$, and deletion of l_i and r_n from the knowledge base.

The transformation to the succeeding rule of the insignificant inference chain in the Fig. 9 is shown in the Fig. 11. A new rule r_5 has been added:

r_5 : *if* f_3 *and* f_6, *then* f_7;

2.1.4 Incorrect Inference Chains

There are two types of incorrect inference chains:

- redundant for input;
- redundant for output.

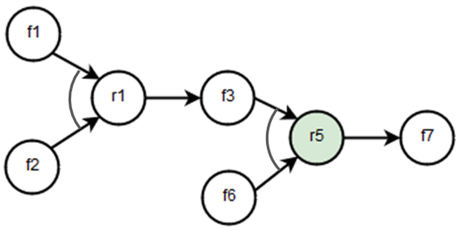

Fig. 11. Transforming to the succeeding rule

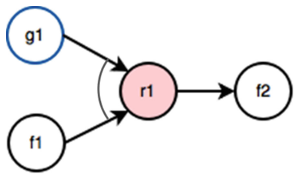

Fig. 12. Inference chain redundant for input

Definition 15. An inference chain l_i is considered as redundant for input one, if $D_{l_i} \cap G \neq \emptyset$.

The Fig. 12 provides an example of a redundant for input inference chain $l_1 = (r_1)$, where:

$r_1 : if\ g_1\ and f_1,\ then f_2;\ g_1 \in G$

The solution is to remove the inference chain redundant for input.

Definition 16. An inference chain l_i is considered as redundant for output one, if $q_{l_i} \in S$.

The Fig. 13. provides an example of a redundant for output inference chain $l_1 = (r_1)$, where:

$r_1 : if\ f_1\ and f_2,\ then\ s_1; s_1 \in S.$

The solution is to remove the inference chain redundant for output one.

2.1.5 Cycles

Definition 17. An inference chain l_i is considered as a cycle, if $q_{l_i} \in D_{l_i}$.

The Fig. 14. provides an example of a cycle $l_1 = (r_1, r_2, r_3)$, where:

$r_1 : if\ f_1\ and f_2,\ then f_3;$
$r_2 : if\ f_3\ and f_4,\ then f_5;$
$r_3 : if\ f_5\ and f_6,\ then f_2;$
$D_{l_1} = \{f_1, f_2\};\ q_{l_1} = \{f_2\};\ q_{l_1} \in D_{l_1};$

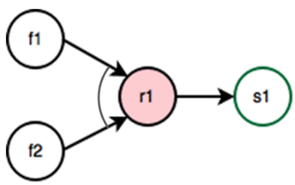

Fig. 13. Inference chain redundant for output

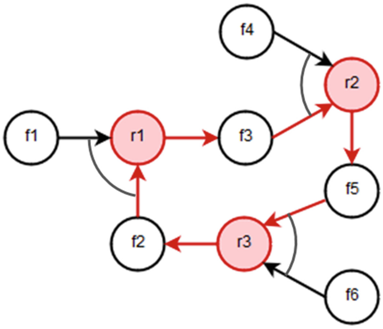

Fig. 14. Cycle

Any cycle discovered in the knowledge base is a structural error, but incorrect rule cannot be selected automatically, so an expert should choose what rule to be removed from the KB.

Definition 18. A rule r_i is considered as a simple cycle, if $q_{r_i} \in D_{r_i}$

The rule r_1 in the Fig. 15 provides an example of a simple cycle:

r_1 : *if* f_1 *and* f_2, *then* f_1;

A simple cycle should be removed from the knowledge base.

2.2 Incompleteness Errors

2.2.1 Isolated Vertices

Definition 19. A vertex $f_i \in F \cup G$ is considered isolated, if $\neg \exists r_j$, where $f_i \in D_{r_j}$ or $f_i \in Q_{r_j}$.

The vertex g_1 in the Fig. 16 is an isolated one, because there is no rule connected to it.

The correction depends on type of the vertex. If the isolated vertex is an input fact or a goal, the solution is to add rules by the expert. Otherwise the vertex should be removed from the knowledge base.

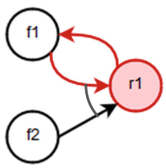

Fig. 15. Simple cycle

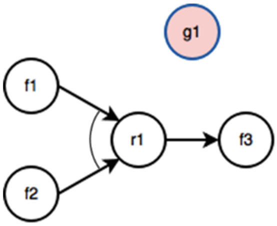

Fig. 16. Isolated vertex

2.3 Inconsistency

C is a set of allowed combinations of facts, so $c_i = \{f_{c_i1}, f_{c_i2}, \ldots, f_{c_in}\}$.

2.3.1 Conflicting Inference Chains

Definition 20. Inference chains l_i and l_j are considered as conflicting ones, if $\exists f_k$, where $f_k \in D_{l_i}$ and $f_k \in D_{l_j}$, and $\neg\exists c_m$, where $q_{l_i} \in c_m$ and $q_{l_j} \in c_m$.

There are inference chains $l_1 = (r_1, r_2)$ and $l_2 = (r_3)$ in the Fig. 17:

$D_{l_1} = \{f_1, f_2\}; q_{l_1} = \{f_7\}$
$D_{l_2} = \{f_2, f_3\}; q_{l_2} = \{f_6\};$

There is no $c_k = (f_6, f_7)$ and $D_{l_1} \cap D_{l_2} = f_2$, so l_1 and l_2 are considered conflicting inference chains.

The correction of conflicting chains error should be done by the expert, either by adding a new allowed combination of facts, or by rewriting or deleting rules.

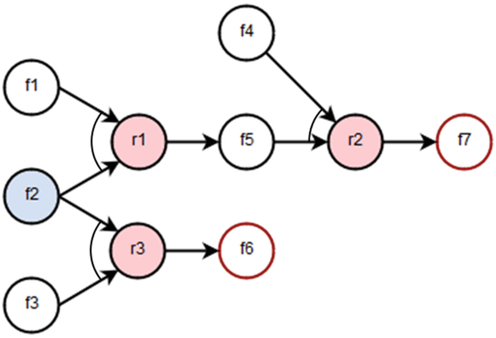

Fig. 17. Conflicting inference chains

3 Conclusion

The paper provides formal classification of structural errors in the knowledge bases of intellectual decision making systems. It describes 3 types of structural errors: redundancy errors, incompleteness errors and inconsistency. For each of group examples are provided. Duplicates, redundant inference chains, insignificant inference chains, incorrect inference chains and cycles are considered as redundancy errors, and isolated vertices error type is incompleteness. Inconsistency in knowledge bases is represented by conflicting inference chains. All errors are formalized in terms of the AND/OR graph model and ways of correction are provided for each defined structural error.

The formal model of structural errors can be used as a basis for performing of automatic verification of rule-based knowledge bases structure.

References

1. Dolinina, O.N.: Razrabotka metoda testirovanija produkcionnyh baz znanij jekspertnyh sistem s uchetom oshibok tipa «zabyvanija ob iskljuchenii»: dis…kand. tehn. nauk/O.N. Dolinina. Saratov, 171 s (1999) (in Russian)
2. Dolinina, O.: Method of the debugging of the knowledge bases of intellectual decision making systems. In: Silhavy, R., Senkerik, R., Oplatkova, Z.K., Silhavy, P., Prokopova, Z. (eds.) Automation Control Theory Perspectives in Intelligent Systems. AISC, vol. 466, pp. 307–314. Springer, Cham (2016). doi:10.1007/978-3-319-33389-2_29
3. Nguen, T., Perkins, W., Laffey, T., Pecors, W.: Checking expert system knowledge bases for consistency and completeness. In: Proceedings of the 9th International Joint Conference on Artificial Intelligence, Los Angeles, pp. 375–378, August 1985
4. Cragun, B.J., Stendel, H.J.: A decision-table-based processor for checking completeness and consistency in rule-based expert systems. Int. J. Man-Mach. Stud. **26**(5), 633–648 (1987)
5. Nguen, T.A.: Verifying consistency of production systems. In: Proceedings of the Third Conference on Artificial Intelligence Applications (CAIA), Kissimmee, Fl, pp. 4–8 (1987)
6. Suwa, M., Scott, A.C., Shortliffe, E.H.: An approach to veryfing consistency and completeness in a rule-based expert system. In: Rule-Based Expert Systems, pp. 159–170. Addison-Wesley, London (1984)
7. Nguen, T., Perkins, W., Laffey, T., Pecora, W.: Checking expert system knowledge bases for consistency and completeness. In: Proceedings of the 9th International Joint Conference on Artificial Intelligence, Los Angeles, pp. 375–378, August 1985