In Proceedings of the 1988 IEEE Workshop on Visual Programming, pp. 45-51

Miró Semantics for Security

Mark W. Maimone J. D. Tygar Jeannette M. Wing

Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

The Miró project at Carnegie Mellon University is designing and implementing a visual language for specifying properties of large software systems. We are designing the language in tandem with giving it a formal semantics. We present the semantics of our language as applied to the security domain.

I Introduction

Visual languages, like all languages, need a formal semantics. This paper presents an outline of a visual language and gives a formal definition of its meaning.

Diagrams from a language that has ambiguous (informal) interpretations for graphical constructs only serve to frustrate the user of the visual language, and confuse the reader ("But what does it *mean*?"). Some languages at least come equipped with rules that determine when a "picture"[1] is well-formed. A formal semantics, however, would describe not only the syntactically valid pictures, but more importantly, their mathematical interpretation. That is, it does not suffice to give only a BNF for pictures; one must additionally map each well-formed picture onto some underlying mathematical entity.

The Miró project at Carnegie Mellon University is designing and implementing a visual language for specifying properties of large software systems. The first class of properties to which we are applying our visual notation is security, e.g., secrecy and integrity of files, as first described in [1]. Unlike the development of many other visual languages, we are designing Miró in tandem with giving it a formal semantics. We therefore benefit from not only the concision of visual notation, but the precision of a rigorous semantics.

II Informal Description

The constructs of Miró are simple: *boxes* and *arrows*, each optionally labeled.[2] In giving such constructs a semantics, it is essential to provide an interpretation for individual boxes and arrows and their compositions when forming *pictures*. Depending on the class of properties of interest, the interpretation of the boxes and arrows will change. In this paper, we present a complete interpretation for the Miró constructs in the domain of security properties. The underlying semantic model for security is simple: an *access-rights* matrix, where each *process* and each *file* (e.g., users and programs) has a (possibly empty) set of *rights* governing access (e.g., Read, Write, Execute). Informally, boxes represent individual processes and files or collections of processes and files; arrows represent rights. Since we have *negative* arrows as well, we can express the absence of rights. The presence of negative arrows introduces some non-triviality to our semantics, since in particular, we wish to disallow ambiguous pictures.

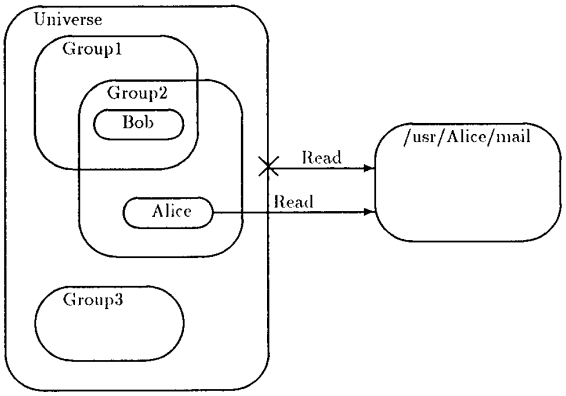

Figure 1: A sample Miró security specification.

For example, Figure 1 shows a Miró security specification that reflects some aspects of the UNIX[3] file protection scheme. The outermost lefthand box depicts a *universe*, Universe, of users, three (out of possibly many not explicitly shown) *groups*, Group1, Group2, and Group3, and two (out of many not explicitly shown) *users*, Alice and Bob. The containment and overlap relationships between the universe, groups, and users indicate that all users are in the universe, and users can be members of more than one group.[4] The righthand box denotes the set of files in Alice's mail directory. The arrows indicate that Alice, and no other user, has Read access to her mail files. That is, the direct positive arrow from Alice overrides the negative arrow from Universe.

Already, the reader might wonder the following: What do "containment" and "overlap" mean when some boxes look like they denote atomic objects (e.g., Alice and Bob) and some denote sets of atomic and non-atomic objects (e.g., Group2)? (What do "atomic" and "non-atomic" mean?) What is the interpretation of the absence of a box or arrow? What is the interpretation of seemingly conflicting negative and positive arrows (i.e., what are the rules for overriding arrows)? How are ambiguous pictures dealt with? These are the sorts of questions a formal semantics can answer precisely.

In what follows, we capture these visual notions in a logical setting: the set theoretic notions hinted at in Figure 1 are made explicit. We formally present the syntactic domains in Section III, and the semantic

[1] We use the term "picture" here generically to mean some ensemble of graphical objects drawn using some visual language.

[2] Boxes are drawn as rectangles with rounded corners, inspired by Harel's Statechart notation [2].

[3] UNIX is a trademark of AT&T.

[4] We do not address the property that a user must belong to at least one group here, though Miró does provide for this expressibility.

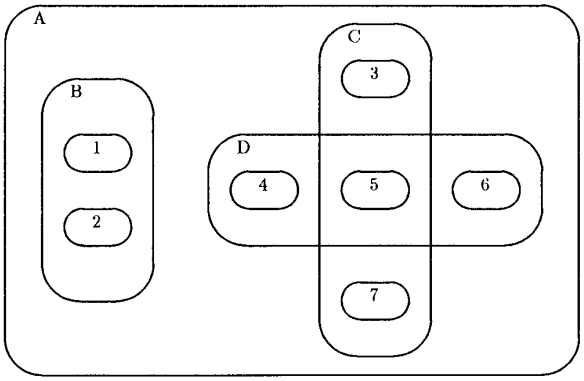

Figure 2: Illustration for the auxiliary definitions. Atomic boxes are labeled with numbers, non-atomic boxes with letters.

X	*members (X)*	*inside (X)*	*contains (X)*	*crisscrosses (X)*
1	{ 1 }	∅	{ 1, A, B }	∅
2	{ 2 }	∅	{ 2, A, B }	∅
3	{ 3 }	∅	{ 3, A, C }	∅
4	{ 4 }	∅	{ 4, A, D }	∅
5	{ 5 }	∅	{ 5, A, C, D }	∅
6	{ 6 }	∅	{ 6, A, D }	∅
7	{ 7 }	∅	{ 7, A, C }	∅
A	{ 1, 2, 3, 4, 5, 6, 7 }	{ 1, 2, 3, 4, 5, 6, 7, B, C, D }	{ A }	∅
B	{ 1, 2 }	{ 1, 2 }	{ A, B }	∅
C	{ 3, 5, 7 }	{ 3, 5, 7 }	{ A, C }	{ D }
D	{ 4, 5, 6 }	{ 4, 5, 6 }	{ A, D }	{ C }

Table 1: Some properties of the picture in Figure 2.

domains (access-rights matrix) and interpretation in Section IV. We present a careful treatment of ambiguity in Section V, and close with some final remarks about current and future work in Section VI.

III Syntax

The basic syntactical elements of a Miró picture are atomic and non-atomic boxes, and typed arrows; these are formally defined in Table 2. For a particular picture, a box in *BOXES* is named by either a process or file identifier. An atomic box in *ATOMS* contains no other box; a non-atomic one contains a set of other boxes. An arrow in *ARROWS* denotes a typed relation between two boxes. The type identifies which kind of access right a process or set of processes (the left hand box) has to a file or set of files (the right hand box). Users are represented by the processes they execute. The only other definition from Table 2 that is of informal interest is τ^* which, for a set of boxes b, defines the union of b and the set of all boxes contained in every box of b.

Throughout the paper, indentation is used to reduce the number of parentheses, negation symbols ($\neg$) will bind more closely than conjunction ($\wedge$), and conjunction will bind more closely than disjunction ($\vee$). The symbol ($\uplus$) will be used to denote the union of two disjoint sets, and ($\triangleq$) will be used to define new constructs.

III.1 Auxiliary Definitions

The following definitions are used in the access matrix specification. They are illustrated in Figure 2 and Table 1. Free variables in the definitions (x, y, P, P', N, N') range over elements in *BOXES*.

members(x): The set of atoms contained in (or equal to) x.

$$members(x) \triangleq \{\ a \mid a \in ATOMS \ \wedge\ a \in \tau^*(\{\ x\ \})\ \}$$

inside(x): The set of boxes whose atoms form a proper subset of those in x. In particular, $x \notin inside\ (x)$. The reader may find it helpful to read $y \in inside\ (x)$ as "box y is strictly inside box x."

$$inside(x) \triangleq \{\ b \mid b \in BOXES \wedge members(b) \subset members(x)\}$$

contains(x): The set of boxes whose atoms are at least those of x. In particular, $x \in contains(x)$. Note that for a given x, *inside (x)* and *contains (x)* are disjoint sets. Read $y \in contains\ (x)$ as "box y is or contains box x."

$$contains(x) \triangleq \{\ b \mid b \in BOXES \wedge members(x) \subseteq members(b)\}$$

crisscrosses (x): The set of boxes which share some, but not all of their atoms with x. Note that this is symmetric, in that $X \in crisscrosses(Y) \ \Rightarrow\ Y \in crisscrosses(X)$. Also note that no box can be both inside and crisscrossing another box (e.g., in Figure 2, box B does not crisscross box A). Read $y \in crisscrosses\ (x)$ as "box y crisscrosses box x (and x crisscrosses y)."

$$crisscrosses(x) \triangleq \{\ b \mid\ b \in BOXES \wedge (members(x) \cap members(b) \neq \emptyset) \wedge (b \notin (inside(x) \uplus contains(x)))\}$$

$x \bowtie y$: x is equal to, or crisscrosses y. Note that this is not a transitive relation. Read $x \bowtie y$ as "box x is equal to or crisscrosses box y."

$$x \bowtie y \triangleq (members(x) = members(y) \vee x \in crisscrosses\ (y))$$

POS $^t(P, P')$: A positive relation of type t exists between P and P'.

Entity	Symbol	Example
Set of Relation Types	$TYPES$	$\{$ *read, write, execute* $\}$
Set of File Identifiers	F_{id}	$\{$ *passwd, vmunix, tty* $\}$
Set of Process Identifiers	P_{id}	$\{$ *alice, bob, charlie* $\}$
Box Constructor:	$B^0_{\mathcal{I}} = \{\langle \emptyset, id\rangle \mid id \in \mathcal{I}\}$	$B^0_{\mathcal{I}}$ is the set of atoms
Set of Boxes with identifiers in $\mathcal{I}$,	$B^1_{\mathcal{I}} = B^0_{\mathcal{I}} \cup \{\langle x, id\rangle \mid x \in (2^{B^0_{\mathcal{I}}} - \{\emptyset\}) \wedge id \in \mathcal{I}\}$	$B^1_{\mathcal{I}}$ is the set of boxes containing atoms
where $\mathcal{I} = F_{id}$ or P_{id}	$B^i_{\mathcal{I}} = \bigcup_{j=0}^{i-1} B^j_{\mathcal{I}} \cup \{\langle x, id\rangle \mid x \in (2^{B^{i-1}_{\mathcal{I}}} - \{\emptyset\}) \wedge id \in \mathcal{I}\}$	$\langle \emptyset, alice\rangle \simeq$ [alice] $\langle \{\langle \emptyset, alice\rangle, \langle \emptyset, bob\rangle\}, students\rangle$
Set of File Boxes	$F \subseteq \bigcup_{i=0}^{\infty} B^i_{F_{id}}$	
Set of Process Boxes	$P \subseteq \bigcup_{i=0}^{\infty} B^i_{P_{id}}$	
Set of All Boxes	$BOXES = F \cup P$	$\{$ [alice], students [alice] [bob] $\}$
Subbox Operators	$\sigma_F : F \to 2^F$ $\sigma_P : P \to 2^P$ $\sigma = \sigma_F \cup \sigma_P$ $\sigma_{\mathcal{I}}(\langle X, id\rangle) = \begin{cases} \emptyset & \text{if } X = \emptyset \\ X & \text{otherwise} \end{cases}$	
	$\tau : 2^{BOXES} \to 2^{BOXES}$ $\tau(X) = \bigcup_{x \in X} \sigma(x)$ $\tau^*(X) = \bigcup_{j=0}^{\infty} \tau^j(\{ X \})$	
Atomic Boxes	$ATOMS = \{x \mid x \in BOXES \wedge \sigma(x) = \emptyset\}$	
Arrows	$ARROWS \subseteq P \times F \times TYPES \times \{ pos, neg \}$	

Table 2: Miró syntactic entities

$POS^t(P, P') \triangleq \langle P, P', t, pos\rangle \in ARROWS$

NEG $^t(N, N')$: A negative relation of type t exists between N and N'.

$NEG^t(N, N') \triangleq \langle N, N', t, neg\rangle \in ARROWS$

To make the interactions of these definitions clearer, we introduce the concept of *box level* and the Closure Lemma. Box level refers to the hierarchy imposed on boxes through containment. Two boxes are said to be *at the same level* if and only if $X \bowtie Y$.[5] If $X \in$ *inside* (Y), Y is said to have a *higher level* than box X, and X a *lower level* than box Y. In Figure 3, A and B have the same level, neither C nor D is related by level to any other box, F has a lower level than E, E has a higher level than F, and F has the same level as itself. It should be noted that box level does *not* provide a partial ordering of boxes.

The following lemma illustrates some relationships among these definitions.

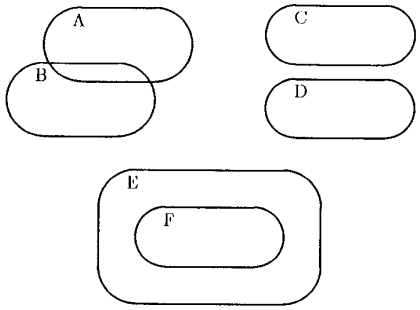

Figure 3: Illustration of *box level*.

Lemma 1 (Closure) *If two boxes B, B' both contain the same atomic box, then exactly one of $B \bowtie B'$, $B \in$* inside *(B'), or $B' \in$* inside *(B) is true.*

Proof We will show first that *at least one* of the conditions holds. Let B, B' be given such that $members(B) \cap members(B') \neq \emptyset$. We will show that the result holds for each of four independent cases, one of which must al-

[5] Just as $\bowtie$ is not transitive, neither is *at the same level*.

ways hold. First suppose $members(B) \subset members(B')$; then by definition of *inside*, $B \in inside\ (B')$. Otherwise, suppose $members(B') \subset members(B)$; then $B' \in inside\ (B)$, also by definition of *inside*. If neither of these holds, suppose $members(B) = members(B')$; then $B \bowtie B'$ by definition of $\bowtie$. Finally, if none of these holds, we know $B \notin contains\ (B')$ since $members(B) \neq members(B')$ and $members(B) \not\subset members(B')$. From that we infer that $B \in crisscrosses\ (B')$ and thus $B \bowtie B'$. Now to see that *at most one* condition will hold, refer to the definitions of $\bowtie$, *inside*, and *crisscrosses*. ♠

IV Semantics

IV.1 Access Rights Matrix

The ultimate interpretation of a Miró picture in the security domain is an access–rights matrix. An access–rights matrix is a standard security entity that represents binary access relations between entities, such as the right for one entity to modify another.

The access–rights matrix Z is three-dimensional, taking a *Process*, a *File*, and a type of *Relation*, with possible values *pos*, *neg*, and *ambig*. The expressions below determine the value of a particular matrix element. Let t be the type of the relation, p an atomic box representing the process, and f an atomic box representing the file. The interpretation is that if $Z\ (p, f, t)$ is *pos* then process p can access file f according to relationship type t. If $Z\ (p, f, t)$ is *neg*, then p cannot access f according to t. If $Z\ (p, f, t)$ is *ambig*, the access cannot be determined. We want to detect and eliminate all such ambiguity in the matrix.

In what follows, P and P' will identify the boxes at the tail and head, respectively, of a positive arrow, and N and N' will identify those at the tail and head of a negative arrow. Boxes that have both kinds of arrows will also have both labels. That is, P and N might label the same box. Boxed symbols (e.g., [x]) are used to name clauses for later reference, and have no semantic or logical interpretation.

$Z(p,f,t)$ is *pos* iff (1)

[A]

$$\exists_{P,P'}\ p \in members(P) \wedge f \in members(P') \wedge POS^t(P, P') \wedge$$
$$\forall_{N,N'}\ p \in members(N) \wedge f \in members(N') \wedge NEG^t(N, N')$$
$$\Rightarrow \neg \left[\begin{array}{l} [1]\ (P \bowtie N \wedge P' \bowtie N') \vee \\ [2]\ N' \in inside\ (P') \vee \\ [3]\ N \in inside\ (P) \end{array} \right]$$

Z is positive when the smallest enclosing boxes have only positive arrows; call these boxes P and P'. We require that no negative arrow join the following pairs of boxes: boxes at the same level as P and P' (case [1] above); one box at a lower level than P or P', and the other box at any level (cases [2] and [3] above).

$Z(p,f,t)$ is *neg* iff (2)

[B]

$$\exists_{N,N'}\ p \in members(N) \wedge f \in members(N') \wedge NEG^t(N, N') \wedge$$
$$\forall_{P,P'}\ p \in members(P) \wedge f \in members(P') \wedge POS^t(P, P')$$
$$\Rightarrow \neg \left[\begin{array}{l} [1]\ (P \bowtie N \wedge P' \bowtie N') \vee \\ [2]\ P' \in inside\ (N') \vee \\ [3]\ P \in inside\ (N) \end{array} \right]$$

$\vee$ [C]

$$\forall_{B,B'}\ B \in contains\ (p) \wedge B' \in contains\ (f) \Rightarrow$$
$$\neg POS^t(B, B') \wedge \neg NEG^t(B, B')$$

Z is negative when the smallest enclosing boxes have only negative arrows (call these boxes N and N'), or when no surrounding boxes are connected by arrows. In the former case, we require that no positive arrow join the following pairs of boxes: boxes at the same level as N and N' (case [1] above); one box at a lower level than N or N', and the other box at any level (cases [2] and [3] above).

$Z(p,f,t)$ is *ambig* otherwise. (3)

The value of an element of Z is ambiguous when neither a positive nor a negative relationship holds. An explicit derivation of those pictures that are ambiguous follows.

IV.2 Uniqueness

Before we derive the explicit conditions for ambiguity, let us first ensure that the other matrix elements are unique. That is, we intend to show that no two atoms can have both a *pos* and *neg* relationship with the same type.

Claim 1 *A relation between two atomic boxes may not be both* pos *and* neg.

Proof Let atomic boxes p and f, and relation type t be given. We will prove the claim by contradiction, using equations (1) and (2) above. Suppose $Z(p, f, t)$ is both *pos* and *neg*. Then [A] (in equation (1)) is true, and from [A] we know there are boxes P and P' containing p and f with a positive arrow connecting them; let us choose such boxes and call them P and P'. We can infer that [C] (in equation (2)) is false because P and P' exist. Thus [B] must be true since we assumed that equation (2) was true. Now from [B] we may choose N and N' containing p and f with a negative arrow connecting them. Using this we can determine that clauses [A1] through [A3] must be false, in order for [A] to be true. Likewise, clauses [B1] through [B3] must be false because P and P' exist and have a positive arrow. Returning to the clauses in [A], we have the following results: by $\neg$ [A1] we know that either $N \not\bowtie P$ or $N' \not\bowtie P'$; by $\neg$ [A2] we know that $N' \notin inside\ (P')$; by $\neg$ [A3] we know that $N \notin inside\ (P)$. Suppose $N \not\bowtie P$. Then by Lemma 1 and $\neg$ [A3], $P \in inside\ (N)$. But $P \notin inside\ (N)$ by $\neg$ [B3], a contradiction. Suppose instead that $N' \not\bowtie P'$. Then by Lemma 1 and $\neg$ [A2], $P' \in inside\ (N')$. But $P' \notin inside\ (N')$ by $\neg$ [B2], a contradiction. ♠

V Ambiguity

We described above explicit conditions for pictures whose contents are known to have a positive or negative relationship. In this section we show that ambiguity can be intuitively defined, derive explicit conditions for pictures whose contents have an ambiguous relationship, and demonstrate these conditions by drawing the corresponding pictures.

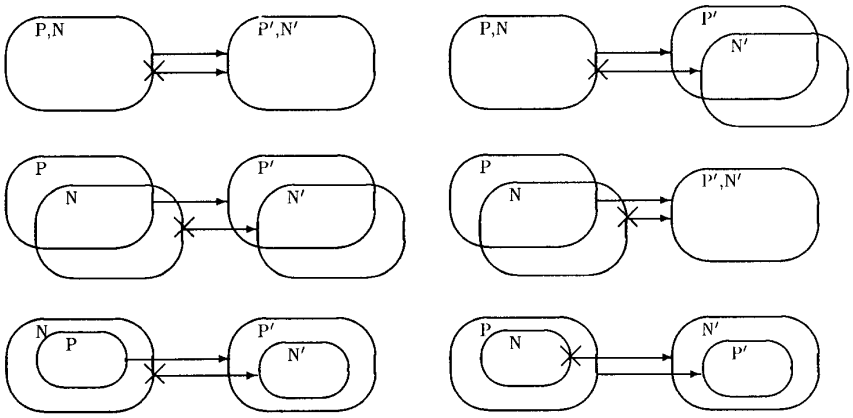

Figure 4: Ambiguous Pictures

V.1 Intuition

We now use the box level concept to state the intuitive definition of ambiguity.

> A relation is *ambiguous* when positive and negative arrows connect pairs of boxes at either the same, or unequal and opposite levels, and at least one of these boxes has the smallest enclosing scope (i.e. no smaller box has a arrow of similar type). These cases are illustrated in Figure 4.

V.2 Derivation

We will show that the intuitive definition of ambiguity given above follows logically from the other definitions. We begin by defining ambiguity as the condition when neither a positive nor a negative relation holds.

$$Z(p,f,t) \text{ is } ambig \text{ iff } \neg(Z(p,f,t) \text{ is } pos) \wedge \neg(Z(p,f,t) \text{ is } neg) \quad (4)$$

We may now use the definitions in section IV.1, and apply one of DeMorgan's laws to the negative case, to expand Equation (4) into the following:

$$\begin{aligned} &Z(p,f,t) \text{ is } ambig \text{ iff} \quad (5)\\ &\neg\exists_{P,P'}\, p \in members(P) \wedge f \in members(P') \wedge POS^t(P,\, P') \wedge\\ &\quad \forall_{N,N'}\, p \in members(N) \wedge f \in members(N') \wedge NEG^t(N,\, N')\\ &\quad\Rightarrow \neg \left[\begin{array}{ll} \boxed{1} & (P \bowtie N \wedge P' \bowtie N') \vee\\ \boxed{2} & N' \in inside\ (P') \vee\\ \boxed{3} & N \in inside\ (P) \end{array}\right]\\ &\quad\wedge\\ &\neg\exists_{N,N'}\, p \in members(N) \wedge f \in members(N') \wedge NEG^t(N,\, N') \wedge\\ &\quad \forall_{P,P'}\, p \in members(P) \wedge f \in members(P') \wedge POS^t(P,\, P')\\ &\quad\Rightarrow \neg \left[\begin{array}{ll} \boxed{1} & (P \bowtie N \wedge P' \bowtie N') \vee\\ \boxed{2} & P' \in inside\ (N') \vee\\ \boxed{3} & P \in inside\ (N) \end{array}\right]\\ &\quad\wedge\\ &\neg\forall_{B,B'}\, B \in contains\ (p) \wedge B' \in contains\ (f) \Rightarrow\\ &\quad \neg POS^t(B,\, B') \wedge \neg NEG^t(B,\, B') \end{aligned}$$

Now we can use properties of first-order logic to push the negations through and express the condition for ambiguity as follows:

$$\begin{aligned} &Z(p,f,t) \text{ is } ambig \text{ iff} \quad (6)\\ &\quad \boxed{\neg A}\\ &\forall_{P,P'}\, \neg(p \in members(P) \wedge f \in members(P') \wedge POS^t(P,\, P')) \vee\\ &\exists_{N,N'}\, p \in members(N) \wedge f \in members(N') \wedge NEG^t(N,\, N') \wedge\\ &\quad \left[\begin{array}{ll} \boxed{1} & (P \bowtie N \wedge P' \bowtie N') \vee\\ \boxed{2} & N' \in inside\ (P') \vee\\ \boxed{3} & N \in inside\ (P) \end{array}\right]\\ &\quad \wedge \boxed{\neg B}\\ &\forall_{N,N'}\, \neg(p \in members(N) \wedge f \in members(N') \wedge NEG^t(N,\, N')) \vee\\ &\exists_{P,P'}\, p \in members(P) \wedge f \in members(P') \wedge POS^t(P,\, P') \wedge\\ &\quad \left[\begin{array}{ll} \boxed{1} & (P \bowtie N \wedge P' \bowtie N') \vee\\ \boxed{2} & P' \in inside\ (N') \vee\\ \boxed{3} & P \in inside\ (N) \end{array}\right]\\ &\quad \wedge \boxed{\neg C}\\ &\exists_{B,B'}\, B \in contains\ (p) \wedge B' \in contains\ (f) \wedge\\ &\quad (POS^t(B,\, B') \vee NEG^t(B,\, B')) \end{aligned}$$

Claim 2 *If the relation between* **p** *and* **f** *is ambiguous according to type* **t**, *then there must be at least two pairs of boxes surrounding both* **p** *and* **f**, *one pair connected by a positive arrow and the other by a negative arrow.*

Proof Suppose the relation between **p** and **f** is ambiguous according to type **t**, i.e., *Z (p, f, t)* is *ambig*. Then equation (6) is true, and in particular the conjuncts labeled $\boxed{\neg A}$ $\boxed{\neg B}$ and $\boxed{\neg C}$ are true. By $\boxed{\neg C}$ we may choose boxes B and B' which have either a positive or negative relation. Suppose the relation is positive. Then by instantiating P and P' to

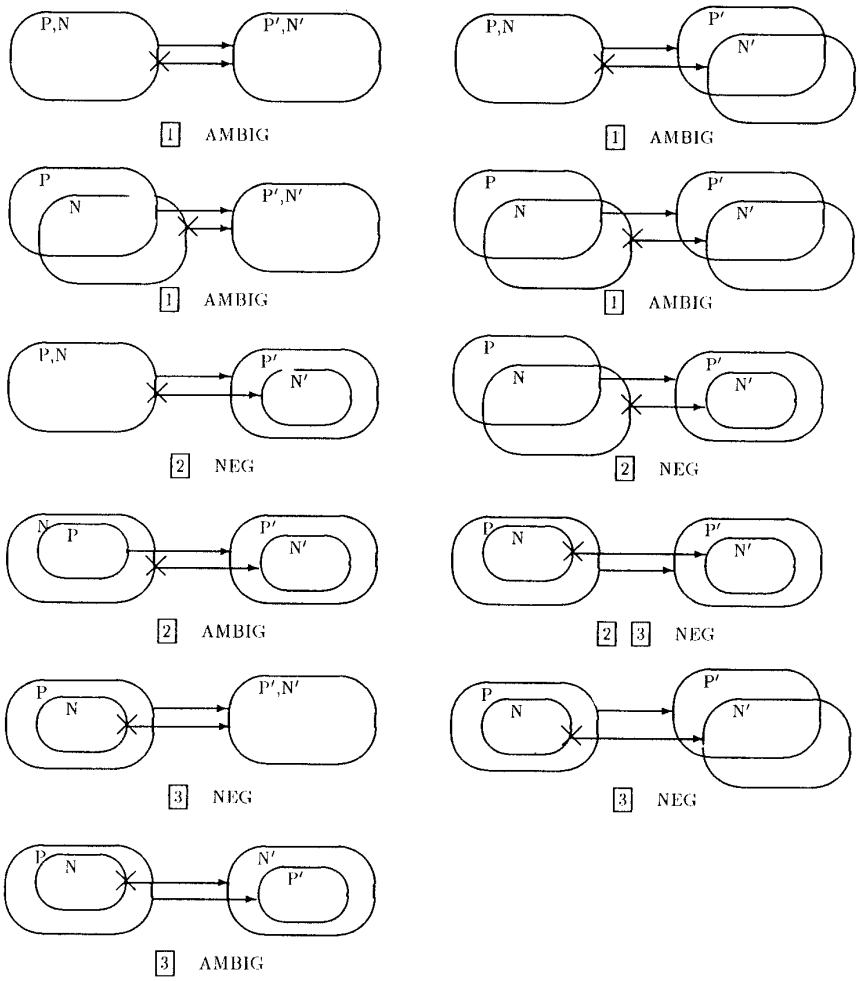

Figure 5: Pictures which contain positive arrows but do not have a positive interpretation

B and B' in $\boxed{\neg A}$ we can derive the existence of boxes with a negative relation. Suppose instead that B and B' have a negative relation. Then by instantiating N and N' to B and B' in $\boxed{\neg B}$ we can derive the existence of boxes with a positive relation. ♠

Figure 5 shows all of the pictures which satisfy the three clauses in $\boxed{\neg A}$, each one labeled with the disjuncts it satisfies and the value it would have according to these semantics. If we now inspect all pictures which satisfy some disjunct in $\boxed{\neg A}$ as well one in $\boxed{\neg B}$, we will find that they are exactly those pictures listed in Figure 4, which are all ambiguous.

VI Conclusions

As hinted in the informal description of the language, Miró also provides the capability to express *constraints* on the relationships depicted within a box and between boxes on abstractions of arrows, called *cables*. These constraints are part of the *type* information of a box. We have an informal interpretation for these constructs and are currently formulating their precise semantics.

Miró solves several important problems in security. It provides, for the first time, a tool for configuring and visualizing complicated security constraints. This tool allows precise definitions of security environments in a convenient mathematical notation — a major advance over previous one-dimensional (i.e., textual) logic-based approaches (such as [3,4,5,6]). It is a practical tool designed to be used by people who are actively enforcing security constraints in real environments.

In the long-term future, we may apply the Miró language to domains outside of security, e.g., concurrency, where we would reinterpret the meaning of boxes and arrows. Ideally, we would like to make that part of the semantics given here that is independent of security a separate library that can be reused by many entities, and reinterpret only that part that is dependent on the specific domain. Finally, we would like to explore the possibility of using a visual approach to giving semantics instead of the standard denotational approach as presented in this paper.

VII Acknowledgments

We thank David Harel for his inspiration and enthusiastic support for our work. We especially wish to thank Amy Moormann, Allan Heydon, and Kenneth McMillan for their comments on earlier versions of this paper.

VIII References

[1] J. D. Tygar and J. M. Wing, "Visual Specification of Security Constraints," in *Proceedings of the 1987 Workshop on Visual Languages*, (Linkoping, Sweden), Aug 1987.

[2] D. Harel, "Statecharts: A Visual Formalism for Complex Systems," *Science of Computer Programming*, vol. 8, 1987.

[3] D. E. Bell and L. J. LaPadula, "Secure Computer Systems: Mathematical Foundations (3 volumes)," Tech. Rep. AD-770 768, AD-771 543, AD-780 528, Mitre Corporation, Nov 1973.

[4] D. Good, R. Cohen, C. Hoch, L. Hunter, and D. Hare, "Report on the Language Gypsy, Version 2.0," Tech. Rep. ICSCA-CMP-10, Certifiable Minicomputer Project, The University of Texas at Austin, September 1978.

[5] T. Benzel, "Analysis of a Kernel Verification," in *Proceedings of the 1984 Symposium on Security and Privacy*, (Oakland, California), pp. 125–131, May 1984.

[6] P. G. Neumann, R. S. Boyer, R. J. Feiertag, K. N. Levitt, and L. Robinson, "A Provably Secure Operating System: The System, Its Applications, and Proofs, Second Edition," Tech. Rep. CSL-116, SRI, May 1980.