Attack–Defense Trees and Two-Player Binary Zero-Sum Extensive Form Games Are Equivalent

Barbara Kordy*, Sjouke Mauw,
Matthijs Melissen**, and Patrick Schweitzer***

University of Luxembourg

Abstract. Attack–defense trees are used to describe security weaknesses of a system and possible countermeasures. In this paper, the connection between attack–defense trees and game theory is made explicit. We show that attack–defense trees and binary zero-sum two-player extensive form games have equivalent expressive power when considering satisfiability, in the sense that they can be converted into each other while preserving their outcome and their internal structure.

1 Introduction

Attack trees [1], as popularized by Bruce Schneier at the end of the 1990s, form an informal but powerful method to describe possible security weaknesses of a system. An attack tree basically consists of a description of an attacker's goal and its refinement into sub-goals. In case of a *conjunctive* refinement, all sub-goals have to be satisfied to satisfy the overall goal, while for a *disjunctive* refinement satisfying any of the sub-goals is sufficient to satisfy the overall goal. The non-refined nodes (i.e., the leaves of the tree) are basic attack actions from which complex attacks are composed.

Due to their intuitive nature, attack trees prove to be very useful in understanding a system's weaknesses in an informal and interdisciplinary context. The development of an attack tree for a specific system may start by building a small tree that is obviously incomplete and describes the attacks at a high level of abstraction, while allowing to refine these attacks and to add new attacks later as to make a more complete description. Over the last few years, attack trees have developed into an even more versatile tool. This is due to two developments. The first development consists of the formalization of the attack trees method [2] which provides an attack tree with a precise meaning. As a consequence, formal analysis techniques were designed [3,4] and computer tools were made commercially available [5,6].

* B. Kordy was supported by the grant No. C08/IS/26 from FNR Luxembourg.
** M. Melissen was supported by the grant No. PHD–09–082 from FNR Luxembourg.
*** P. Schweitzer was supported by the grant No. PHD–09–167 from FNR Luxembourg.

T. Alpcan, L. Buttyan, and J. Baras (Eds.): GameSec 2010, LNCS 6442, pp. 245–256, 2010.

The second development comes from the insight that a more complete description can be achieved by modeling the activities of a system's defender in addition to those of the attacker. Consequently, one can analyze which set of defenses is optimal from the perspective of, for instance, cost effectiveness. Several notions of protection trees or defense nodes have already been proposed in the literature [7,8]. They mostly consist of adding one layer of defenses to the attack tree, thus ignoring the fact that in a dynamic system new attacks are mounted against these defenses and that, consequently, yet more defenses are brought into place. Such an alternating nature of attacks and defenses is captured in the notion of attack–defense trees [9]. In this recently developed extension of attack trees, the iterative structure of attacks and defenses can be visualized and evolutionary aspects can be modeled.

These two developments, the formalization of attack trees and the introduction of defenses, imply that an attack–defense tree can be formally considered as a description of a game. The purpose of this paper is to make the connection between attack–defense trees and game theory explicit. We expect that the link between the relatively new field of attack modeling and the well-developed field of game theory can be exploited by making game theoretic analysis methods available to the attack modeling community. As a first step, we study the relation between attack–defense trees and games in terms of expressiveness. Rather than studying the graphical attack–defense tree language, we consider an algebraic representation of such trees, called *attack–defense terms* (ADTerms) [9], which allows for easier formal manipulation.

The main contribution of this paper is to show that ADTerms with a satisfiability attribute are equivalent to two-player binary zero-sum extensive form games. Whenever we talk about games, we refer to a game in this class. We show equivalence by defining two mappings: one from games to ADTerms and one from ADTerms to games. Then, we interpret a *strategy* in the game as a *basic assignment* for the corresponding ADTerm and vice versa. Such a basic assignment expresses which attacks and defenses are in place. Equivalence then roughly means that for every winning strategy, there exists a basic assignment that yields a satisfiable term, and vice versa. Although the two formalisms have much in common, their equivalence is not immediate. Two notions in the domain of ADTerms have no direct correspondence in the world of games: conjunctive nodes and refinements. The mapping from ADTerms into games will have to solve this in a semantically correct way.

This paper is structured as follows. We introduce attack–defense terms and two-player binary zero-sum extensive form games in Section 2. In Section 3 we define a mapping from games to attack–defense terms and prove that a player can win the game if and only if he is successful in the corresponding ADTerm. A reverse mapping is defined in Section 4.

Proofs of theorems are not included due to space restrictions, and can be found in a technical report [10].

2 Preliminaries

2.1 Attack–Defense Trees

A limitation of attack trees is that they cannot capture the interaction between attacks carried out on a system and defenses put in place to fend off the attacks. To mitigate this problem and in order to be able to analyze an attack–defense scenario, attack–defense trees are introduced in [9]. Attack–defense trees may have two types of nodes: attack nodes and defense nodes, representing actions of two opposing players. The attacker and defender are modeled in a purely symmetric way. To avoid differentiating between attack–defense scenarios with an attack node as a root and a defense node as a root, the notions of *proponent* (denoted by p) and *opponent* (denoted by o) are introduced. The root of an attack–defense tree represents the main goal of the proponent. To be more precise, when the root is an attack node, the proponent is an attacker and the opponent is a defender, and vice versa.

To formalize attack–defense trees we use attack–defense terms. Given a set S, we write S^* for the set of all strings over S and ε for the empty string.

Definition 1. *Attack–defense terms (ADTerms) are typed ground terms over a signature $\Sigma = (S, F)$, where*

- $S = \{\mathrm{p}, \mathrm{o}\}$ *is a set of types (we denote $-\mathrm{p} = \mathrm{o}$ and $-\mathrm{o} = \mathrm{p}$),*
- $F = \{(\vee_k^{\mathrm{p}})_{k\in\mathbb{N}}, (\wedge_k^{\mathrm{p}})_{k\in\mathbb{N}}, (\vee_k^{\mathrm{o}})_{k\in\mathbb{N}}, (\wedge_k^{\mathrm{o}})_{k\in\mathbb{N}}, \mathrm{c}^{\mathrm{p}}, \mathrm{c}^{\mathrm{o}}\} \cup \mathbb{B}^{\mathrm{p}} \cup \mathbb{B}^{\mathrm{o}}$ *is a set of functions equipped with a mapping* $\mathrm{type}\colon F \to S^* \times S$*, which expresses the type of each function as follows. For $k \in \mathbb{N}$,*

$$\begin{aligned}
&\mathrm{type}(\vee_k^{\mathrm{p}}) = (\mathrm{p}^k, \mathrm{p}) && \mathrm{type}(\vee_k^{\mathrm{o}}) = (\mathrm{o}^k, \mathrm{o})\\
&\mathrm{type}(\wedge_k^{\mathrm{p}}) = (\mathrm{p}^k, \mathrm{p}) && \mathrm{type}(\wedge_k^{\mathrm{o}}) = (\mathrm{o}^k, \mathrm{o})\\
&\mathrm{type}(\mathrm{c}^{\mathrm{p}}) = (\mathrm{po}, \mathrm{p}) && \mathrm{type}(\mathrm{c}^{\mathrm{o}}) = (\mathrm{op}, \mathrm{o})\\
&\mathrm{type}(b) = (\varepsilon, \mathrm{p}),\ \textit{for } b \in \mathbb{B}^{\mathrm{p}} && \mathrm{type}(b) = (\varepsilon, \mathrm{o}),\ \textit{for } b \in \mathbb{B}^{\mathrm{o}}.
\end{aligned}$$

The elements of $\mathbb{B}^{\mathrm{p}}$ and $\mathbb{B}^{\mathrm{o}}$ are typed constants, which represent basic actions of the proponent and the opponent, respectively. The functions $\vee_k^{\mathrm{p}}, \wedge_k^{\mathrm{p}}, \vee_k^{\mathrm{o}}, \wedge_k^{\mathrm{o}}$ represent disjunctive ($\vee$) and conjunctive ($\wedge$) refinement operators of arity k, for a proponent (p) and an opponent (o), respectively. Whenever it is clear from the context, we omit the subscript k. The binary function c^s ('counter'), where $s \in S$, connects a term of the type s with a countermeasure. By T_Σ we denote the set of all ADTerms. We partition T_Σ into T_Σ^{p} (the set of terms of the proponent's type) and T_Σ^{o} (the set of terms of the opponent's type). To denote the type of a term, we define a function $\tau\colon T_\Sigma \to S$ by $\tau(t) = s$ if $t \in T_\Sigma^s$.

Example 1. The ADTerm $t = \mathrm{c}^{\mathrm{p}}(\wedge^{\mathrm{p}}(E, F), \vee^{\mathrm{o}}(G)) \in T_\Sigma^{\mathrm{p}}$ is graphically displayed in Fig. 1 (left). For this ADTerm, we have $\tau(t) = \mathrm{p}$. Subterms E and F are basic actions of the proponent's type, and G is a basic action of the opponent's type.

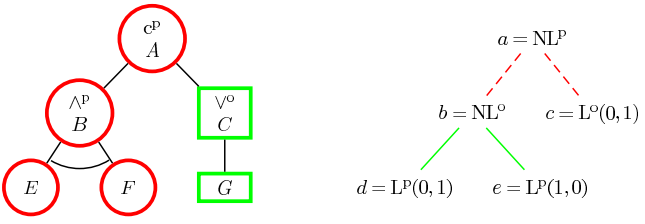

Fig. 1. An example of an ADTerm (left) and a two-player binary zero-sum extensive form game (right)

Assuming the proponent is the attacker, this means that the system can be attacked by combining the basic attack actions E and F. However the defender has the option to defend if he implements the basic defense action G.

In order to check whether an attack–defense scenario is feasible, we introduce the notion of satisfiability of an ADTerm by defining a satisfiability attribute sat. First, for player $s \in \{\mathrm{p}, \mathrm{o}\}$ we define a *basic assignment* for s as a function $\beta^s \colon \mathbb{B}^s \to \{true, false\}$. We gather the basic assignments for both players in a *basic assignment profile* $\beta = (\beta^{\mathrm{p}}, \beta^{\mathrm{o}})$. Second, the function $\mathrm{sat} \colon T_\Sigma \to \{true, false\}$ is used in order to calculate the satisfiability value of an ADTerm. It is defined recursively as follows

$$\mathrm{sat}(t) = \begin{cases} \beta^s(t^s), & \text{if } t = t^s \in \mathbb{B}^s, \\ \vee(\mathrm{sat}(t_1), \ldots, \mathrm{sat}(t_k)), & \text{if } t = \vee^s(t_1, \ldots, t_k), \\ \wedge(\mathrm{sat}(t_1), \ldots, \mathrm{sat}(t_k)), & \text{if } t = \wedge^s(t_1, \ldots, t_k), \\ \mathrm{sat}(t_1) \wedge \neg\, \mathrm{sat}(t_2), & \text{if } t = \mathrm{c}^s(t_1, t_2). \end{cases}$$

For instance, consider the term t from Example 1 and the basic assignment profile $\beta = (\beta^{\mathrm{p}}, \beta^{\mathrm{o}})$, where $\beta^{\mathrm{p}}(E) = true$, $\beta^{\mathrm{p}}(F) = true$, $\beta^{\mathrm{o}}(G) = false$. We get $\mathrm{sat}(t) = true$. Assuming the proponent is the attacker, this means that the basic defense action G is absent and the system is attacked by combining the basic attack actions E and F.

The next definition formalizes the notion of a satisfiable ADTerm for a player.

Definition 2. *For every player s, strategy β^s and strategy profile β, we define the sets of ADTerms $\mathrm{Sat}^s_\beta, \mathrm{Sat}^s_{\beta^s}, \mathrm{Sat}^s \subseteq T_\Sigma$ in the following way. Let $t \in T_\Sigma$.*

- *$t \in \mathrm{Sat}^s_\beta$ if either $\tau(t) = s$ and $\mathrm{sat}(t) = true$, or $\tau(t) = -s$ and $\mathrm{sat}(t) = false$. In this case we say that s is successful in t under β.*
- *$t \in \mathrm{Sat}^s_{\beta^s}$ if $t \in \mathrm{Sat}^s_{(\beta^{\mathrm{p}}, \beta^{\mathrm{o}})}$ for every basic assignment β^{-s}. In this case we say that s is successful in t under β^s.*
- *$t \in \mathrm{Sat}^s$ if there exists a basic assignment β^s for player s such that $t \in \mathrm{Sat}^s_{\beta^s}$. In this case we say that t is satisfiable for s.*

Theorem 1. *For every ADTerm t, we have that every basic assignment profile β partitions T_Σ into* $\mathrm{Sat}^{\mathrm{p}}_\beta$ *and* $\mathrm{Sat}^{\mathrm{o}}_\beta$.

Proof. This follows immediately from the first item in Definition 2.

2.2 Two-Player Binary Zero-Sum Extensive Form Games

We consider *two-player binary zero-sum extensive form games*, in which a proponent p and an opponent o play against each other. In those games, we allow only for the outcomes $(1,0)$ and $(0,1)$, where $(1,0)$ means that the proponent succeeds in his goal (breaking the system if he is the attacker, keeping the system secure if he is the defender), and $(0,1)$ means that the opponent succeeds. Note that the proponent is not necessarily the player who plays first in the game. Finally, we restrict ourselves to extensive form games, i.e., games in tree format. Our presentation of games differs from the usual one, because we present games as terms. This eases the transformation of games into ADTerms. We formalize games in the next definition, where L stands for a leaf and NL for a non-leaf of the term.

Definition 3. *Let $S = \{\mathrm{p}, \mathrm{o}\}$ denote the set of players and* $\mathrm{Out} = \{(1,0),(0,1)\}$ *the set of possible outcomes. A two-player binary zero-sum extensive form game is a term $t ::= \psi^{\mathrm{p}} \mid \psi^{\mathrm{o}}$, where*

$$\psi^{\mathrm{p}} ::= \mathrm{NL}^{\mathrm{p}}(\psi^{\mathrm{o}}, \ldots, \psi^{\mathrm{o}}) \mid \mathrm{L}^{\mathrm{p}}(1,0) \mid \mathrm{L}^{\mathrm{p}}(0,1)$$
$$\psi^{\mathrm{o}} ::= \mathrm{NL}^{\mathrm{o}}(\psi^{\mathrm{p}}, \ldots, \psi^{\mathrm{p}}) \mid \mathrm{L}^{\mathrm{o}}(1,0) \mid \mathrm{L}^{\mathrm{o}}(0,1).$$

We denote the set of all two-player binary zero-sum extensive form games by $\mathcal{G}$. We define the first player *of a game ψ^s as the function $\tau\colon \mathcal{G} \to S$ such that $\tau(\psi^s) = s$.*

Example 2. An example of a two-player binary zero-sum extensive form game is the expression $\mathrm{NL}^{\mathrm{p}}(\mathrm{NL}^{\mathrm{o}}(\mathrm{L}^{\mathrm{p}}(0,1), \mathrm{L}^{\mathrm{p}}(1,0)), \mathrm{L}^{\mathrm{o}}(0,1))$. This game is displayed in Fig. 1 (right). When displaying extensive form games, we use dashed edges for choices made by the proponent, and solid edges for those made by the opponent. In this game, first the proponent can pick from two options; if he chooses the first option, the opponent can choose between outcomes $(0,1)$ and $(1,0)$. If the proponent chooses the second option, the game will end with outcome $(0,1)$.

Definition 4. *A function σ^s is a* strategy *for a game $g \in \mathcal{G}$ for player $s \in S$ if it assigns to every non-leaf of player s in g $\mathrm{NL}^s(\psi_1^{-s}, \ldots, \psi_n^{-s})$ a term ψ_k^{-s} for some $k \in \{1, \ldots, n\}$.*

A strategy profile *for a game $g \in \mathcal{G}$ is a pair $\sigma = (\sigma^{\mathrm{p}}, \sigma^{\mathrm{o}})$, where σ^{p} is a strategy of g for* p*, and σ^{o} a strategy of g for* o.

If $g = \mathrm{NL}^s(\psi_1^{-s}, \ldots, \psi_n^{-s})$ and $\sigma = (\sigma^{\mathrm{p}}, \sigma^{\mathrm{o}})$, sometimes we abuse notation and write $\sigma(g) = \psi_k^{-s}$ where $\psi_k^{-s} = \sigma^s(g)$.

Now we define the outcome of a game in three steps.

Definition 5. *We say that* $(0,1) \leq^{\mathrm{p}} (1,0)$ *and* $(1,0) \leq^{\mathrm{o}} (0,1)$*, so that* $(\mathrm{Out}, \leq^{\mathrm{p}})$ *and* $(\mathrm{Out}, \leq^{\mathrm{o}})$ *are totally ordered sets. Let* $(r^{\mathrm{p}}, r^{\mathrm{o}})$ *be an element of* Out*, and* $\psi_1^{-s}, \ldots, \psi_n^{-s}$ *be games with player* $-s$ *as the first player.*

1. *The* outcome $\mathrm{out}_{(\sigma^{\mathrm{p}},\sigma^{\mathrm{o}})} \colon \mathcal{G} \to \mathrm{Out}$ *of a game* g *under strategy profile* $\sigma = (\sigma^{\mathrm{p}}, \sigma^{\mathrm{o}})$ *is defined by:*

$$\begin{aligned} \mathrm{out}_{(\sigma^{\mathrm{p}},\sigma^{\mathrm{o}})}(\mathrm{L}^s(r^{\mathrm{p}}, r^{\mathrm{o}})) &= (r^{\mathrm{p}}, r^{\mathrm{o}}) \\ \mathrm{out}_{(\sigma^{\mathrm{p}},\sigma^{\mathrm{o}})}(\mathrm{NL}^s(\psi_1^{-s}, \ldots, \psi_n^{-s})) &= \mathrm{out}_{(\sigma^{\mathrm{p}},\sigma^{\mathrm{o}})}(\sigma^s(\mathrm{NL}^s(\psi_1^{-s}, \ldots, \psi_n^{-s}))) \end{aligned}$$

2. *The* outcome $\mathrm{out}_{\sigma^s} \colon \mathcal{G} \to \mathrm{Out}$ *of a game* g *under strategy* σ^s *is defined by:*

$$\begin{aligned} \mathrm{out}_{\sigma^s}(\mathrm{L}^s(r^{\mathrm{p}}, r^{\mathrm{o}})) &= (r^{\mathrm{p}}, r^{\mathrm{o}}) \\ \mathrm{out}_{\sigma^s}(\mathrm{NL}^s(\psi_1^{-s}, \ldots, \psi_n^{-s})) &= \mathrm{out}_{\sigma^s}(\sigma^s(\mathrm{NL}^s(\psi_1^{-s}, \ldots, \psi_n^{-s}))) \\ \mathrm{out}_{\sigma^s}(\mathrm{NL}^{-s}(\psi_1^{-s}, \ldots, \psi_n^{-s})) &= \max_{1 \leq i \leq n}{}^{\leq^{-s}} \{\mathrm{out}_{\sigma^s}(\psi_i^{-s})\} \end{aligned}$$

3. *The* outcome $\mathrm{out} \colon \mathcal{G} \to \mathrm{Out}$ *of a game* g *is defined by:*

$$\begin{aligned} \mathrm{out}(\mathrm{L}^s(r^{\mathrm{p}}, r^{\mathrm{o}})) &= (r^{\mathrm{p}}, r^{\mathrm{o}}) \\ \mathrm{out}(\mathrm{NL}^s(\psi_1^{-s}, \ldots, \psi_n^{-s})) &= \max_{1 \leq i \leq n}{}^{\leq^{s}} \{\mathrm{out}_{\sigma^s}(\psi_i^{-s})\} \\ \mathrm{out}(\mathrm{NL}^{-s}(\psi_1^{-s}, \ldots, \psi_n^{-s})) &= \max_{1 \leq i \leq n}{}^{\leq^{-s}} \{\mathrm{out}_{\sigma^s}(\psi_i^{-s})\} \end{aligned}$$

Here $\mathrm{out}_{(\sigma^{\mathrm{p}},\sigma^{\mathrm{o}})}$ denotes the outcome of the game when p and o play according to strategy σ^{p} and σ^{o}, respectively. Furthermore out_{σ^s} denotes the outcome if player s plays strategy σ^s, and player $-s$ tries to achieve the best possible outcome for himself. Finally, out denotes the outcome of the game if both players try to maximize their own outcome.

3 From Games to ADTerms

In this section, we show how to transform binary zero-sum two-player extensive form games into ADTerms. We define a function that transforms games into ADTerms, and a function that transforms a strategy for a game into a basic assignment for the corresponding ADTerm. First we show that the player who wins the game is also the player for whom the corresponding ADTerm is satisfiable, if both players play the basic assignment corresponding to their strategy in the game. Then we show that if a player has a strategy in a game which guarantees him to win, he is successful in the corresponding ADTerm under the corresponding basic assignment. For this purpose, we first define a function $[\cdot]_{\mathrm{AD}}$ that maps games into ADTerms.

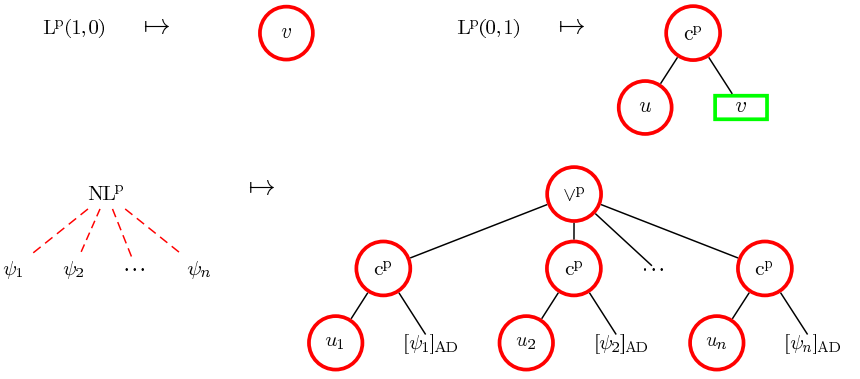

Fig. 2. Transformation of a game in extensive form into an ADTerm by function $[\cdot]_{\mathrm{AD}}$

Definition 6. *Let v^s, u^s, and $u^s_1, \ldots, u^s_n$, for $s \in S$, represent fresh basic actions from $\mathbb{B}^s$. The function $[\cdot]_{\mathrm{AD}} \colon \mathcal{G} \to T_\Sigma$ is defined in the following way.*

$$\mathrm{L}^{\mathrm{p}}(1,0) \mapsto v^{\mathrm{p}} \tag{1a}$$
$$\mathrm{L}^{\mathrm{o}}(1,0) \mapsto \mathrm{c}^{\mathrm{o}}(u^{\mathrm{o}}, v^{\mathrm{p}}) \tag{1b}$$
$$\mathrm{L}^{\mathrm{p}}(0,1) \mapsto \mathrm{c}^{\mathrm{p}}(u^{\mathrm{p}}, v^{\mathrm{o}}) \tag{1c}$$
$$\mathrm{L}^{\mathrm{o}}(0,1) \mapsto v^{\mathrm{o}} \tag{1d}$$
$$\mathrm{NL}^{\mathrm{p}}(\psi_1, \ldots, \psi_n) \mapsto \vee^{\mathrm{p}}(\mathrm{c}^{\mathrm{p}}(u^{\mathrm{p}}_1, [\psi_1]_{\mathrm{AD}}), \ldots, \mathrm{c}^{\mathrm{p}}(u^{\mathrm{p}}_n, [\psi_n]_{\mathrm{AD}})) \tag{1e}$$
$$\mathrm{NL}^{\mathrm{o}}(\psi_1, \ldots, \psi_n) \mapsto \vee^{\mathrm{o}}(\mathrm{c}^{\mathrm{o}}(u^{\mathrm{o}}_1, [\psi_1]_{\mathrm{AD}}), \ldots, \mathrm{c}^{\mathrm{o}}(u^{\mathrm{o}}_n, [\psi_n]_{\mathrm{AD}})). \tag{1f}$$

The rules for player p are visualized in Fig. 2 (the rules for player o are symmetric). The rules specify that a winning leaf for a player in the game is transformed into a satisfiable ADTerm for this player, i.e., an ADTerm consisting of only a leaf belonging to this player (Rule (1a)–(1d)), and that non-leaves in the game are transformed into disjunctive ADTerms of the same player (Rule (1e)–(1f)). These disjunctions have children of the form $\mathrm{c}^s(u^{\mathrm{p}}_k, [\psi_k]_{\mathrm{AD}})$ for some k. The intended meaning here is that player s selects u^{p}_k exactly when his strategy selects ψ_k in the game. An example of a transformation of a game into an ADTerm is depicted in Fig. 3.

The resulting ADTerm is thus conjunction-free. Note that because terms in games alternate between p and o, this procedure results in valid ADTerms (i.e., in terms of the form $\mathrm{c}^s(u^{s_1}, v^{s_2})$, $s_1 = s$ and $s_2 = -s$, and disjunctive terms for player s have children for player s as well).

Now we define how to transform a strategy profile for a game into a basic assignment profile for an ADTerm. First we define a transformation $[\![\cdot]\!]_{\mathrm{AD}}$ from a strategy σ^s ($s \in \{\mathrm{p}, \mathrm{o}\}$) for game g into a basic assignment $\beta^s = [\![\sigma^s]\!]_{\mathrm{AD}}$ for ADTerm $[g]_{\mathrm{AD}}$. Intuitively, if a player's strategy for the game selects a certain branch, the basic assignment for the ADTerm assigns *true* to the node u_k in the corresponding branch, and *false* to the nodes u_k in the other branches. Furthermore, ADTerms resulting from leaves in the game are always selected.

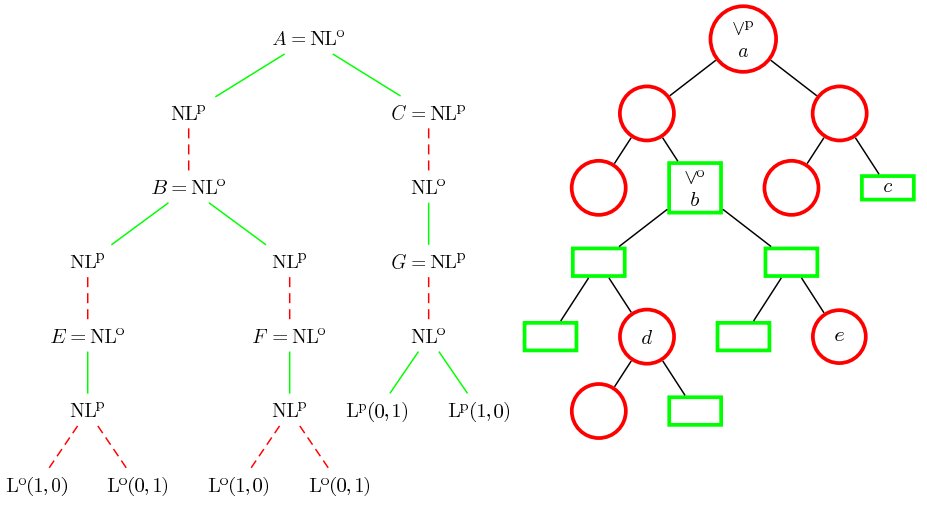

Fig. 3. The result of the transformation of the ADTerm from Fig. 1 into a game (left), and the game from Fig. 1 into an ADTerm (right)

Definition 7. *Let s be a player, g be a game and σ^s be a strategy of player s for g. The function $\beta^s = [\![\sigma^s]\!]_{\mathrm{AD}}$ is defined as follows. For all ADTerms $\mathrm{c}^s(u^s, v^{-s})$ and v^s resulting from the first four cases in Definition 6, we set $\beta^s(u^s) = \beta^s(v^s) = true$. For ADTerms obtained from game g by one of the last two cases in Definition 6, if $\sigma^s(g) = \psi_k$, we set $\beta^s(u^s_k) = true$ and $\beta^s(u^s_i) = false$ for $1 \leq i \leq n$, $i \neq k$.*

The strategy profile $(\beta^{\mathrm{p}}, \beta^{\mathrm{o}})$ can be transformed into a basic assignment profile by $[\![(\beta^{\mathrm{p}}, \beta^{\mathrm{o}})]\!]_{\mathrm{AD}} = ([\![\beta^{\mathrm{p}}]\!]_{\mathrm{AD}}, [\![\beta^{\mathrm{o}}]\!]_{\mathrm{AD}})$.

The next theorem states that a player is the winner in a game under a certain strategy profile if and only if he is successful in the corresponding ADTerm under the basic assignment profile corresponding to the strategy profile.

Theorem 2. *Let g be a game and σ a strategy profile for g. Then $\mathrm{out}_\sigma(g) = (1,0)$ if and only if $[g]_{\mathrm{AD}} \in \mathrm{Sat}^{\mathrm{p}}_{[\![\sigma]\!]_{\mathrm{AD}}}$.*

The following theorem states that a strategy in a game guarantees player s to win if and only if s is successful in the corresponding ADTerm under the corresponding basic assignment. Surprisingly, this is not a consequence of Theorem 2: there might be a basic assignment β^s for the ADTerm, for which there exists no strategy σ^s such that $\beta^s = [\![\sigma^s]\!]_{\mathrm{AD}}$ (i.e, the function $[\![\cdot]\!]_{\mathrm{AD}}$ is not surjective). Therefore it is not immediately clear that if a player has a strategy σ^s that wins from the other player independent of his strategy, a player with a basic assignment $[\![\sigma^s]\!]_{\mathrm{AD}}$ wins from the other player independent of his basic assignment.

Theorem 3. *Let g be a game and σ^{p} be a strategy for* p *on g. Then* $\mathrm{out}_{\sigma^{\mathrm{p}}}(g) = (1,0)$ *if and only if* $[g]_{\mathrm{AD}} \in \mathrm{Sat}^{\mathrm{p}}_{[\![\sigma^{\mathrm{p}}]\!]_{\mathrm{AD}}}$.

From this theorem, we immediately get the following corollary.

Corollary 1. *Whenever g is a game,* $\mathrm{out}(g) = (1,0)$ *if and only if* $[g]_{\mathrm{AD}} \in \mathrm{Sat}^{\mathrm{p}}$.

4 From ADTerms to Games

We proceed with the transformation in the other direction. We define two transformations, namely from ADTerms into games, and from basic assignment profiles into strategy profiles. Then we show that if a player has a basic assignment for an ADTerm with which he is successful, the corresponding strategy in the corresponding game guarantees him to win.

Definition 8. *We define a function $[\cdot]_{\mathrm{G}}$ from ADTerms to games as follows:*

$$v^{\mathrm{p}} \mapsto \mathrm{NL}^{\mathrm{o}}(\mathrm{NL}^{\mathrm{p}}(\mathrm{L}^{\mathrm{o}}(0,1), \mathrm{L}^{\mathrm{o}}(1,0))) \tag{2a}$$

$$v^{\mathrm{o}} \mapsto \mathrm{NL}^{\mathrm{p}}(\mathrm{NL}^{\mathrm{o}}(\mathrm{L}^{\mathrm{p}}(1,0), \mathrm{L}^{\mathrm{p}}(0,1))) \tag{2b}$$

$$\vee^{\mathrm{p}}(\psi_1, \ldots, \psi_n) \mapsto \mathrm{NL}^{\mathrm{o}}(\mathrm{NL}^{\mathrm{p}}([\psi_1]_{\mathrm{G}}, \ldots, [\psi_n]_{\mathrm{G}})) \tag{2c}$$

$$\vee^{\mathrm{o}}(\psi_1, \ldots, \psi_n) \mapsto \mathrm{NL}^{\mathrm{p}}(\mathrm{NL}^{\mathrm{o}}([\psi_1]_{\mathrm{G}}, \ldots, [\psi_n]_{\mathrm{G}})) \tag{2d}$$

$$\wedge^{\mathrm{p}}(\psi_1, \ldots, \psi_n) \mapsto \mathrm{NL}^{\mathrm{o}}(\mathrm{NL}^{\mathrm{p}}([\psi_1]_{\mathrm{G}}), \ldots, \mathrm{NL}^{\mathrm{p}}([\psi_n]_{\mathrm{G}})) \tag{2e}$$

$$\wedge^{\mathrm{o}}(\psi_1, \ldots, \psi_n) \mapsto \mathrm{NL}^{\mathrm{p}}(\mathrm{NL}^{\mathrm{o}}([\psi_1]_{\mathrm{G}}), \ldots, \mathrm{NL}^{\mathrm{o}}([\psi_n]_{\mathrm{G}})) \tag{2f}$$

$$\mathrm{c}^{\mathrm{p}}(\psi_1, \psi_2) \mapsto \mathrm{NL}^{\mathrm{o}}(\mathrm{NL}^{\mathrm{p}}([\psi_1]_{\mathrm{G}}), [\psi_2]_{\mathrm{G}}) \tag{2g}$$

$$\mathrm{c}^{\mathrm{o}}(\psi_1, \psi_2) \mapsto \mathrm{NL}^{\mathrm{p}}(\mathrm{NL}^{\mathrm{o}}([\psi_1]_{\mathrm{G}}), [\psi_2]_{\mathrm{G}}) \tag{2h}$$

A graphical representation of the rules for player p is displayed in Fig. 4 (the rules for player o are symmetric). It can easily be checked that this construction guarantees valid games (in which p-moves and o-moves alternate). According to these rules, we transform leaves for player s into two options for player s, a losing and a winning one (Rules (2a) and (2b)). These choices correspond to not choosing and choosing the leaf in the ADTerm, respectively. Disjunctive terms for player s are transformed into choices for player s in the game (Rules (2c) and (2d)). There is no direct way of representing conjunctions in games. We can still handle conjunctive terms though, by transforming them into choices for the other player (Rules (2e) and (2f)). This reflects the fact that a player can succeed in all his options exactly when there is no way for the other player to pick an option which allows him to succeed. Finally, countermeasures against player s are transformed into a choice for player $-s$ (Rules (2g) and (2h)). Here, the first option corresponds to player $-s$ not choosing the countermeasure, so that it is up to player s whether he succeeds or not, while the second option corresponds to player $-s$ choosing the countermeasure. The transformation of a game into an ADTerm is illustrated in Fig. 3.

We proceed by defining a transformation $[\![\cdot]\!]_{\mathrm{G}}$ from a basic assignment for an ADTerm into a strategy for the corresponding game. We only give the definition for $s = \mathrm{p}$; the definition for $s = \mathrm{o}$ is symmetric.

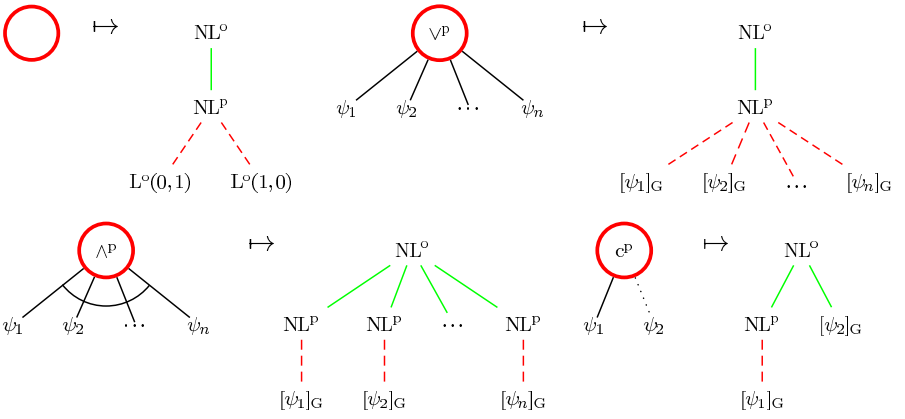

Fig. 4. Transformation of an ADTerm into a game by means of function $[\cdot]_{\mathrm{G}}$

Definition 9. *Function* $[\![\cdot]\!]_{\mathrm{G}}$ *is a transformation from a basic assignment* β^{p} *for ADTerm* t *into a strategy* $\sigma^{\mathrm{p}} = [\![\beta^{\mathrm{p}}]\!]_{\mathrm{G}}$ *for the game* $[t]_{\mathrm{G}}$*. If a (sub)term from* $[t]_{\mathrm{G}}$ *is obtained by rule* $(2n)$ *in Definition 8, then* σ^{p} *of that (sub)term is defined by rule* $(3n)$ *in this definition.*

$$\sigma^{\mathrm{p}}(\mathrm{NL}^{\mathrm{p}}(\mathrm{L}^{\mathrm{o}}(0,1),\mathrm{L}^{\mathrm{o}}(1,0))) = \mathrm{L}^{\mathrm{o}}(1,0) \quad \text{if } \beta^s(v) = \mathit{true}. \tag{3a}$$
$$= \mathrm{L}^{\mathrm{o}}(0,1) \quad \text{otherwise.}$$
$$\sigma^{\mathrm{p}}(\mathrm{NL}^{\mathrm{p}}([\psi_1]_{\mathrm{G}},\ldots,[\psi_n]_{\mathrm{G}})) = [\psi_k]_{\mathrm{G}} \tag{3c}$$
where k *is the smallest number such that* $\psi_k \in \mathrm{Sat}^{\mathrm{p}}_{\beta^{\mathrm{p}}}$.
$$= [\psi_1]_{\mathrm{G}}$$
if there exists no such number.
$$\sigma^{\mathrm{p}}(\mathrm{NL}^{\mathrm{p}}(\mathrm{NL}^{\mathrm{o}}([\psi_1]_{\mathrm{G}}),\ldots,\mathrm{NL}^{\mathrm{o}}([\psi_n]_{\mathrm{G}}))) = \mathrm{NL}^{\mathrm{o}}([\psi_k]_{\mathrm{G}}) \tag{3f}$$
where k *is the smallest number such that* $\psi_k \in \mathrm{Sat}^{\mathrm{p}}_{\beta^{\mathrm{p}}}$.
$$= \mathrm{NL}^{\mathrm{o}}([\psi_1]_{\mathrm{G}})$$
if there exists no such number.
$$\sigma^{\mathrm{p}}(\mathrm{NL}^{\mathrm{p}}(\mathrm{NL}^{\mathrm{o}}([\psi_1]_{\mathrm{G}}),[\psi_2]_{\mathrm{G}})) = \mathrm{NL}^{\mathrm{o}}([\psi_1]_{\mathrm{G}}) \text{ if } \psi_2 \notin \mathrm{Sat}^{\mathrm{p}}_{\beta^{\mathrm{p}}}. \tag{3h}$$
$$= [\psi_2]_{\mathrm{G}} \quad \text{otherwise.}$$

For Rules (3b), (3d), (3e) and (3g), σ^{p} *is trivially defined as there is only one refinement.*

Note that some of the rules, namely Rules (3c), (3f) and (3h), are non-local in the sense that we need to evaluate all subterms of the ADTerm before we can decide what to play in the game.

Theorem 4. *Let* t *be an ADTerm and* β^{p} *a basic assignment for* t*. Then* $t \in \mathrm{Sat}^{\mathrm{p}}_{\beta^{\mathrm{p}}}$ *if and only if* $\mathrm{out}_{[\![\beta^{\mathrm{p}}]\!]_{\mathrm{G}}}([t]_{\mathrm{G}}) = (1,0)$.

Now we obtain immediately the following corollary by definition of out and $\mathrm{Sat}^{\mathrm{p}}$.

Corollary 2. *Whenever* t *is an ADTerm,* $t \in \mathrm{Sat}^{\mathrm{p}}$ *if and only if* $\mathrm{out}([t]_{\mathrm{G}}) = (1,0)$.

5 Conclusion

We showed that attack–defense terms and binary zero-sum two-player extensive form games have equivalent expressive power when considering satisfiability, in the sense that they can be converted into each other while preserving their outcome. Moreover, the transformations preserved internal structure, in the sense that there exists injections between subterms in the game and subterms in the ADTerm such that if a player wins in the subterm of the game, the corresponding subterm in the ADTerm is satisfiable for this player, and vice versa. Therefore attack–defense trees with a satisfiability attribute and binary zero-sum two-player extensive form games can be seen as two different representations of the same concept. Both representations have their advantages. On the one hand, attack–defense trees are more intuitive, because conjunctions and refinements can be explicitly modeled. On the other hand, the game theory representation profits from the well-studied theoretical properties of games.

We saw that two notions in the domain of ADTerms had no direct correspondence to notions in the world of games: conjunctive nodes and refinements. The first problem has been solved by transforming conjunctive nodes for one player to disjunctive nodes for the other player. This also shows that, when considering the satisfiability attribute, the class of conjunction-free ADTerms has equal expressive power to the full class of ADTerms (note that the transformation from ADTerms into games and vice versa are not each other's inverse, i.e., $[[t]_{\mathrm{AD}}]_{\mathrm{G}} \neq t$ and $[[t]_{\mathrm{G}}]_{\mathrm{AD}} \neq t$). The second problem has been solved by adding extra dummy moves with only one option for the other player in between refining and refined nodes.

In the future, we plan to consider attack–defense trees accompanied with more sophisticated attributes, such that a larger class of games can be converted. An example of these are non-zero-sum games, where $(1, 1)$ can be interpreted as an outcome where both the attacker and the defender profit (for example, if the attacker buys his goal from the defender), and $(0, 0)$ as an outcome where both parties are damaged (when the attacker fails in his goal, but his efforts damage the defender in some way). Also the binary requirement can be lifted, so that the outcome of a player represents for instance the cost or gain of his actions. Furthermore, it would be interesting to look for a correspondence of incomplete and imperfect information in attack–defense trees.

Acknowledgments. The authors would like to thank Leon van der Torre and Wojciech Jamroga for valuable discussions on the topic of this paper.

References

1. Schneier, B.: Secrets and lies. Wiley, Indianapolis (2004)
2. Mauw, S., Oostdijk, M.: Foundations of Attack Trees. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 186–198. Springer, Heidelberg (2005)
3. Willemson, J., Jürgenson, A.: Serial Model for Attack Tree Computations. In: Lee, D., Hong, S. (eds.) ICISC 2009. LNCS, vol. 5984, pp. 118–128. Springer, Heidelberg (2010)

4. Rehák, M., Staab, E., Fusenig, V., Pěchouček, M., Grill, M., Stiborek, J., Bartoš, K., Engel, T.: Runtime Monitoring and Dynamic Reconfiguration for Intrusion Detection Systems. In: Kirda, E., Jha, S., Balzarotti, D. (eds.) RAID 2009. LNCS, vol. 5758, pp. 61–80. Springer, Heidelberg (2009)
5. Amenaza: SecurITree, http://www.amenaza.com/
6. Isograph: AttackTree+, http://www.isograph-software.com/atpover.htm
7. Edge, K.S., Dalton II, G.C., Raines, R.A., Mills, R.F.: Using Attack and Protection Trees to Analyze Threats and Defenses to Homeland Security. In: MILCOM, pp. 1–7. IEEE, Los Alamitos (2006)
8. Bistarelli, S., Dall'Aglio, M., Peretti, P.: Strategic Games on Defense Trees. In: Dimitrakos, T., Martinelli, F., Ryan, P.Y.A., Schneider, S. (eds.) FAST 2006. LNCS, vol. 4691, pp. 1–15. Springer, Heidelberg (2006)
9. Kordy, B., Mauw, S., Radomirović, S., Schweitzer, P.: Foundations of Attack–Defense Trees. In: FAST. LNCS. Springer, Heidelberg (2010), http://satoss.uni.lu/members/barbara/papers/adt.pdf
10. Kordy, B., Mauw, S., Melissen, M., Schweitzer, P.: Attack–defense trees and two-player binary zero-sum extensive form games are equivalent – technical report with proofs, http://arxiv.org/abs/1006.2732