Organizing Computational Problem Solving Communities via Semantic Games

A dissertation presented

by

Ahmed Abdelmeged

to the Faculty of the Graduate School

of the College of Computer and Information Science

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Northeastern University

Boston, Massachusetts

February, 2014

ABSTRACT

Competitions have been used to build and organize communities aiming to solve complex computational problems. A recent successful computational biology contest, hosted on TopCoder with a prize pool of only $6000, has attracted 122 contestants that eventually produced an algorithm that is both more accurate and is 1,000 times faster than the best corresponding algorithm developed by the National Institute of Health.

Our take on state of the art computational problem solving competitions is that they follow the contest pattern where participants submit their algorithms to be evaluated by a *trusted* administrator. The administrator is completely responsible for conducting a correct and thorough evaluation of all submitted algorithms. Depending on the computational problem of interest, it can be a massive undertaking for the administrator to conduct such an evaluation. For example, consider the effort involved in collecting benchmarks for the SAT competitions.

By resorting to a peer evaluation approach, the administrator is no longer required to undertake the task of evaluating all submitted algorithms. However, there is no guarantee that all submitted algorithms are correctly and thoroughly evaluated. On one hand, thoroughness of evaluation can be expected to improve as more participants get involved in peer evaluation. Also, thoroughness of evaluation can be assessed with generic code coverage tools. On the other hand, it becomes harder to ensure the correctness of evaluation as more participants are involved in peer evaluation.

It more critical for administrators to ensure correct evaluation. Thoroughness code coverage / larger community / admin involvement.

For example, some participants may evaluate algorithms provided by their peers using test cases with the wrong output.

A competition is essentially an evaluation process that reveals information

about the knowledge of their participants about the underlying problem domain. This information can be rewarding for some participants. Also, participants can also use this information to figure out which other participants they may learn from. A competition sponsor can also use this information to figure out the amount of reward to give to participants.

Fairness is a highly desirable quality for competitions. A fair competition reveals *accurate* information about the knowledge of its participants. Fairness is an attractive property for participants who are willing to invest their effort to be rewarded based on the competition results. Fairness is also important for a competition sponsor that is interested in having the *best* solution known among the community of participants. Therefore,

State of the art

State of the art computational problem competitions follow the contest pattern where a *trusted* administrator *fairly* evaluates all submitted solutions to determine the best among them. The main disadvantage of this pattern is that the administrator is completely responsible for implementing a correct and *convincingly* thorough evaluation process. Depending on the computational problem of interest, it can be a massive undertaking for the administrator to implement such an evaluation process. For example, consider the effort involved in collecting benchmarks for the SAT competitions.

On one hand, delegating the responsibility of evaluating participants to their competitors can reduce the evaluation overhead on the administrator. On the other hand, certain participants may be more thoroughly evaluated than others. Furthermore, some participants can be unfairly evaluated. For example, their submitted algorithms may be evaluated using a test case with the wrong output.

A Semantic Game (SG) is a constructive debate of the truth of some logical claim between two distinguished parties: the verifier which asserts that the claim holds, and the falsifier which asserts that the claim does not hold. An SG can be used to *fairly* evaluate computational problem solutions with minimal administrator overhead, yet under quite limiting conditions.

Although it is a property of SGs that one of its participants must have a winning strategy, it remains fair to *demerit* the loser if it voluntarily chose their side. The rationale is that the loser either did not select the correct side (i.e. the side with the winning strategy) or selected the correct side but did not play according to the winning strategy. An SG can serve as a simple, generic and fair algorithm evaluation process in a competition with two participants provided that one participant voluntarily takes the verifier side and the other participant voluntarily takes the falsifier site. Essentially, an SG is used to debate the existence of an algorithm that satisfies some logical specification. As a part of playing the SG, the verifier is required to submit an algorithm satisfying that logical specification. The benefits of this simple SG-based algorithm evaluation process is that it involves participants, reduces the evaluation overhead on the administrator yet remains fair.

Semantic Games (SGs) of interpreted predicate logic statements can serve as a fair evaluation process for computational problem solutions with minimal

this creates a conflict of interest situation where participants are incentivised to disqualify their competitor's solutions. For example, participants may evaluate algorithms submitted by their competitors using test cases with the wrong output. participants may submit test cases with the wrong output to evaluate algorithms submitted by their competitors.

Increasing participants' involvement in the evaluation process of algorithms can make the evaluation process more convincing and reduce the evaluation overhead on the administrator. Unfortunately, this creates a conflict of interest situation

where participants are incentivised to disqualify algorithms submitted by others. For example, participants may submit test cases with the wrong output to evaluate algorithms submitted by other participants. Therefore, it is critical to have enough checks and balances to ensure that submitted algorithms are *fairly* evaluated by peer participants.

TABLE OF CONTENTS

Chapter Page

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

Competitions have been successfully used to build and organize communities aiming to solve complex computational problems. A recent success of this paradigm is demonstrated by the following quote from a recent Nature publication [3]: "Historically, prize-based contests have had striking success in attracting unconventional individuals who can overcome difficult challenges. To determine whether this approach could solve a real big-data biologic algorithm problem, we used a complex immunogenomics problem as the basis for a two-week online contest broadcast to participants outside academia and biomedical disciplines. Participants in our contest produced over 600 submissions containing 89 novel computational approaches to the problem. Thirty submissions exceeded the benchmark performance of the US National Institutes of Health's MegaBLAST. The best achieved both greater accuracy and speed (1,000 times greater)."

There are numerous other examples of competitions organized to encourage the research and development on computational problem solving. Examples include the SAT competition [2] organized to encourage the development of high performance SAT solvers. Examples also include software development competitions held on platforms such as TopCoder [5]. There are also numerous other examples of software development competitions organized for educational purposes such as competitions held on platforms such as Project Euler [1] and Jutge [4].

The result of a computational problem solving competition is either a rating or a ranking of the problem solving skills of the competition's participants. A community aiming to solve a computational problem can benefit from the result of a competition held between its members in a number of ways. First, to motivate community members to develop skills related to solving the underlying computational problem; the competition result is often used to objectively distribute a prize. Even with no prize to distribute, the announcement of competition results can be rewarding to some participants. Second, to effectively diffuse problem solving knowledge. The knowledge of top ranked participants can be shared with other community members to learn from before the next competition starts. This is called "leveling-the-boats" and is quite common. Finally, sponsors may be interested in the competition results in order to hire top ranked participants or to use the solutions they provide during competitions. Sponsors may offer a prize to attract more high quality members to the community.

It is crucial that the competition result accurately reflects the skill level of participants in solving the underlying computational problem. Otherwise, the competition can fail to attract a community and sponsors.

State of the art computational problem solving competitions follow the contest pattern where a *trusted* administrator is responsible for preparing an objective, correct and thorough process to evaluate the skills of individual participants. Depending on the computational problem of interest, it can be a massive undertaking for the administrator to prepare such evaluation process. For example, consider the effort involved in collecting 2.86 Gigabytes worth of compressed benchmarks for the SAT 2013 competition [2].

Luckily, a sports-like approach may be used to organize computational problem solving competitions with minimal overhead on the administrator. In sports, participants peer-evaluate their opponents. The administrator's role, typically called a referee in sports contexts, is to ensure that participants follow a set of *easily checkable* rules. This sports-like approach was used in the historic competition held, in 1535, between Tartaglia and Fior to figure out who knows how to solve cubic equations more efficiently. The rules were that each provides the other with 30 cubic equations to solve and the faster wins. One imagines it was a relatively easy task to ensure that equations supplied by both Tartaglia and Fior were indeed cubic equations and solutions produced by both Tartaglia and Fior were indeed correct.

Unfortunately, direct adoption of this approach to more complex problems may defeat the purpose of reducing the overhead on the administrator. For example, consider the following two-party hypothetical competition to develop a correct SAT solver. Each party provides 30 CNF formulas to be solved by the SAT solver developed by their opponent. In this competition, the administrator would be required to check the correctness of solutions produced by both SAT solvers. When one of the solvers claims a particular CNF formula to be unsatisfiable, the administrator has to check that the CNF formula is indeed unsatisfiable. It is however a much harder task than it is to check the correctness of some claimed solution to some cubic equation.

In this dissertation, we develop a sports-like approach to organize computational problem solving competitions with a minimal overhead on the administrator. In the rest of this chapter, we present thesis summarizing our approach. Then we summarize of our contributions. Finally we describe the organization of this dissertation.

1.1 Thesis Statement and Rationale

Our thesis is: “Semantic games of interpreted logic sentences provide a useful foundation to organize computational problem solving competitions.”. Below, we illustrate the rationale behind this thesis.

A Semantic Game (SG) is a constructive debate of the truth of some interpreted logic sentence between two distinguished parties: the verifier which asserts that the claim holds, and the falsifier which asserts that the claim does not hold. The rules of an SG are systematically derived from the syntax of the underlying claim. An SG gives a meaning to its underlying claim in the sense that the underlying claim is $true$ (respectively $false$) if and only if there is a winning strategy for the verifier (respectively falsifier).

An SG can be used to evaluate the problem solving skills related to a particular computational problem with a minimal overhead on the administrator. Furthermore, this evaluation is guaranteed to never underestimate the skills of participants. In a nutshell, an SG can be used to constructively debate an interpreted logic sentence formally specifying a particular computational problem. It is correct to demerit the problem solving skills, related to the computational problem specified by the underlying logical sentence, of the loser of an SG because the loser must have either picked the wrong side or not played according to the winning strategy.

We now illustrate our approach by an example. Consider the following interpreted logical sentence specifying the MAXimum SATisfiability (MAX-SAT) problem: $\forall \phi \in CNFs \; \exists v \in assignments(\phi) \forall f \in assignments(\phi). \; fsat(f,\phi) \leq fsat(v,\phi)$. This logical sentence is interpreted in a structure that defines all non-logical symbols; namely, $CNFs$, $assignments$, $fsat$ and $\leq$. Before an SG can be played, the two participants pick their sides in the debate. For now, we assume they picked opposite sides. An SG played on this logical sentence proceeds as follows:

1. the falsifier provides a CNF formula ϕ.
2. given ϕ, the verifier provides an assignment v for the variables in ϕ.
3. given ϕ and v, the falsifier provides an assignment f for the variables in ϕ.

The verifier wins if the assignement v it provided satisfies at least as many clauses as those satisfied by the assignement f provided by the falsifier $fsat(f,\phi) \leq fsat(v,\phi)$.

We now use the rules of the SG to illustrate few points mentioned earlier:

- An SG can be used to evaluate the problem solving skills related to a particular computational problem; in order to win, participants must exercise skills related to solving the computational problem specified by the underlying logical statement. by analyzing the rules we may conclude that it is in the best interest of the verifier to provide an assignment v that satisfies the maximum satisfiable fraction of clauses. We also may conclude that it is in the best interest of the falsifier to provide an assignement f satisfying more than v does, when possible. It is also in the best interest of the falsifier to provide a CNF formula ϕ where it is hard for the verifier to find the assignment satisfying the maximum fraction of satisfiable clauses.
- SG-based evaluation can be carried out with a minimal overhead on the administrator; The administrator is essentially responsible for implementing the structure in which the underlying logical statement is interpreted. It is always possible to rewrite the underlying logical formula scraping some of the quantifiers either in or out of the structure and consequently adding or reducing overhead on the administrator. In our example, the administrator only has to check whether a given formula ϕ is indeed a correct CNF formula, check whether an assignment is indeed a correct assignment for the variables in a given formula, compute the fraction of clauses of some CNF formula

satisfied by some assignment, and compare two such fractions. All of these are relatively easy tasks for the administrator to perform. We may rewrite the underlying formula scraping the right most quantifier into the structure $\forall \phi \in CNFs\ \exists v \in assignments(\phi).\ max - sat(\phi, v)$. By doing so, the administrator becomes responsible for the much harder task of checking whether an assignment is indeed satisfying the maximum satisfiable fraction in a given CNF formula.

- SG-based evaluation is guaranteed to never underestimate the skills of participants; losing an SG can always be blamed on lacking skills related to solving the computational problem specified by the underlying logical problem. In our example, the underlying logical sentence of our example is indeed correct. Therefore, there is a winning strategy for the verifier; namely, to provide an assignment v satisfying the maximum possible fraction of clauses in any formula it may be given. Should the verifier lose, it can be blamed for not providing an assignment satisfying the maximum possible fraction of satisfiable clauses. Should the falsifier lose, it can be blamed for picking the wrong side in the debate.
- – [overestimation blamed on opponent.]
- – [constructive - learning.]
- – [constructive - outcome.]

1.2 Key Challenge and Contributions

In this dissertation, we develop a sports-like approach to organize computational problem solving competitions with a minimal overhead on the administrator. As we argued earlier, SGs can achieve this goal, yet only for two-party competitions. We overcome this, rather severe, restriction by using a tournament of SGs. However, we carefully design our tournament such that the tournament as a whole retains those desirable properties of the two-party SG-based evaluation especially those properties related to accurate estimation of skills of participants.

Collusion between participants is a key challenge to designing a SG tournament that guarantees that the skills of participants are never underestimated. A colluding participant may choose to lose SGs on purpose in order to inflate the estimated skill of another specific participant. We managed to successfully overcome this challenge by developing a provably collusion resilient ranking function for SG tournaments. Furthermore, we developed formal characterization of collusion resilient ranking functions that provides a practical entry point to designers willing to develop other collusion resilient ranking functions for SG tournaments.

We claim the following contributions:

1. We propose the idea of organizing computational problem solving competitions using tournaments of SGs.

2. We develop a simplified version of SGs in order to make them more accessible.

3. We develop a theory of collusion resilient ranking of SG tournaments.

1.3 Organization

The rest of this dissertation is organized as follows: In Chapter 2 we present background information on SGs and how they can be utilized to organize communities around computational problems.

Chapter 2

Background: Semantic Games

What are semantic games and how they are played. Different kinds of logic have different kinds of Semantic Games. There are non-standard semantic games (with retractable moves)?

Chapter 3

Organizing Computational Problem Solving Communities

The purpose of this chapter is to concretely describe our approach to organizing computational problem solving communities. We designed a kind of structured interaction spaces for computational problem solving communities. We call it a *lab*. A lab formally defines a computational problem using an interpreted predicate logic sentence. Community members interact in a lab through a competition consisting of several semantic games of the interpreted predicate logic sentence defining the lab's computational problem.

participating in semantic games of the

organize a community aiming to solve labs as an interaction spaces for communities: Related concepts are Wikipedia pages and TopCoder marathon matches. A lab has a well defined claim family. participants contribute to labs by ?submitting avatars capable of playing simplified sgs?.

Labs:

Defining a lab: Formulation: Sec 2.2 Expression: Sample lab definitions:

Participating in a lab:

Chapter 4

Collusion Resilient Ranking of Semantic Game Tournaments

4.1 Approach at a Glance

Suppose that we have an underlying *true* claim C specifying a computational problem. Therefore, there must be a winning strategy for a verifier in any SG of C. Suppose that p_1 and p_2 are two perfectly acting participants. Therefore, both will choose to take the verifier side in an SG of C. Also, both p_1 and p_2 will apply the winning strategy when taking the verifier side in an SG of C. Table 4.1 shows the outcome of a double round robin semantic game tournament between p_1 and p_2. p_1 and p_2 play two SGs on C. In the first game, shown in the top right cell, p_2 takes the verifier side and p_1 is forced to take the falsifier side. In the second game, shown in the bottom left cell, p_1 takes the verifier side and p_2 is forced to take the falsifier side. By virtue of being perfectly acting participants, p_2 wins the first game and p_1 wins the second game.

We demonstrate our notion of collusion resilient ranking using the following four score-based ranking approaches:

Fal \ Ver	p_1	p_2
p_1	-	p_2
p_2	p_1	-

Table 4.1: Outcome of a Semantic Game Tournament with Two Perfectly Acting Participants

- Number of wins (#W): For each participant, we count the number of wins for every participant throughout the tournament. The higher the participant scores, the better the participant's rank is.
- Number of losses (#L): For each participant, we count the number of losses for every participant throughout the tournament. The lower the participant scores, the better the participant's rank is.
- Number of wins against a non forced participant (#WNF): we only count the number of games a participant has won against a non forced participant. A participant is said to be forced in an SG if it takes the opposite side to the side the participant chooses to take. The higher the participant score, the better the participant's rank is.
- Number of faults (#NFL): we only count the number of games a participant loses while taking its chosen side. The lower the participant scores, the better the participant's rank is.

Both #WNF and #NFL ignore games in which the loser is forced as a form of compensation for players at a disadvantage. The rational is that in such games it could be that the loser had no chance of winning whatsoever. More specifically, this happens when the opponent of the forced loser is a non forced perfectly acting player. But, unfortunately, it is not always computationally tractable to decide

Participant	#W	#L	#WNF	#NFL
p_1	1	1	0	0
p_2	1	1	0	0

Table 4.2: Evaluating a Semantic Game Tournament with Two Perfectly Acting Participants

Fal \ Ver	p_1	p_2	p_3
p_1	-	p_2	p_3
p_2	p_1	-	p_2
p_3	p_1	p_2	-

Table 4.3: Outcome of a Semantic Game Tournament with Two Colluding Participants

whether a player is indeed choosing the correct side or is employing the winning strategy. Therefore, both approaches ignore all games in which the loser is forced.

Table 4.2 demonstrates the scores of p_1 and p_2 according to the four ranking approaches. Each participant wins a single game. Therefore, both score a single point using the #W approach. Each participant loses a single game. Therefore, both score a single point using the #L approach. Each participant wins only against a forced player. Therefore, both score zero points using the #WNF approach. Each participant loses only while forced. Therefore, both score zero points using the #NFL approach. Using either of the four ranking approaches, both p_1 and p_2 are top ranked.

Now, suppose that a third player p_3 has joined the tournament not for the purpose of competing with p_1 and p_2 for the top rank, but to cut p_1 short from being top ranked. We assume p_3 has access to the winning strategy of p_2 and will use it except against p_2. This situation is illustrated in Table 4.3. The highlighted cell marks the semantic game that p_3 loses on purpose for the benefit of p_2.

Now, we examine the four ranking approaches to determine which ones are

Participant	#W	#L	#WNF	#NFL
p_1	2	2	0	0
p_2	3	1	1	0
p_3	1	3	0	1

Table 4.4: Evaluating a Semantic Game Tournament with Two Colluding Participants

resilient to the collusion between p_3 and p_2. Table 4.4 shows the scores of p_1, p_2 and p_3 using the four ranking approaches. For each ranking approach, the cells corresponding to the best scores are highlighted. Among the four ranking approaches we examined, we note that fault counting is the only collusion resilient approach.

The rest of this chapter provides a formal, in depth study of collusion resilient ranking functions. In Section 4.2, we formalize the notions of beating functions representing tournament results as well as ranking functions. Then, in Section 4.3, we formalize our notion of collusion resilient ranking functions. We show that it is impossible to have a collusion resilient ranking function that also have the desirable property of encouraging winning. The natural consequence of this impossibility is the need to relax the requirement of encouraging wins to never discouraging losses when combining it with the collusion resilience requirement.

We then give an alternative characterization for the space of collusion resilient ranking functions that never discourage winning nor encourage losing. The alternative characterization provides a more accessible entry point to the design of collusion resilient ranking functions. In Section 4.4 we give two complete characterizations of fault counting by adding extra axioms to the characterization of the aforementioned space of collusion resilient ranking functions.

In Section 4.5 we informally discuss few alternative ranking functions. We then conclude this chapter in Section 4.6 with a discussion of related work.

4.2 Formalizing Beating and Ranking Functions

In this section we formalize the notions of beating and ranking functions. We use a beating function to represent a tournament result and use a ranking function to produce an ordering of tournament participants based on a tournament result. We also describe the algebraic structure of beating functions. In subsequent sections, we rely on the operations in this structure to formulate properties of beating and ranking functions and use the laws governing these operations in our proofs.

4.2.1 Notation

We modeled our notation for variables after the Hungarian notation used to name variables in computer programs. The goal is to avoid as many quantifiers as possible when expressing logical formulas. For example, instead of writing $\forall p \in P.\ \Phi(p)$, we directly write $\Phi(p)$. Also, instead of writing $\forall a,b \in P.\ \Phi(a,b)$ we directly write $\Phi(p_a, p_b)$.

Explicitly, we use a single capital Latin letter to denote a set and use the same small Latin letter to denote an element of the set. Subscripts are used to distinguish multiple elements of the same set, when necessary. Constants are denoted using boldface font. Functions are denoted using small Latin letters and superscripts are used to denote their type parameters. Free variables are assumed to be universally quantified.

In our proofs, we put labels to the right of formulas. We use the notation $LABEL[term_1/var_1, \ldots term_n/var_n]$ to denote a particular instantiation of the formula labeled $LABEL$ in which the freely occurring variables $var_1, \ldots var_n$ are replaced with the terms $term_1, \ldots term_n$ respectively.

4.2.2 Beating Functions

Let $\mathbf{s_v}$ and $\mathbf{s_f}$ be two constants denoting the verifier and falsifier sides respectively. Let $\mathbf{S} = \{\mathbf{s_v}, \mathbf{s_f}\}$. We use a **beating** function $b^p : P \times P \times \mathbf{S} \times \mathbf{S} \times \mathbf{S} \to \mathbb{N}$ to represent the results of all semantic games comprising a tournament among a finite set of players P. $b^p(p_w, p_l, s_{wc}, s_{lc}, s_w)$ denotes the number of semantic games won by p_w against p_l where p_w chooses to take the side s_{wc} and p_l chooses to take the side s_{lc} and s_w is the actual side taken by the p_w. We use B^P to denote the set of all possible beating functions for a given finite set P of players.

4.2.3 Ranking Functions

We define a ranking to be a reflexive, transitive and complete binary relation. We use R^P to denote the set of all possible rankings of a given set P of players. A **ranking function** $\preceq : B^P \to R^P$ associates some ranking to every beating function. We say that p_x is at least as good as p_y according to the ordering assigned by the ranking function $\preceq$ to the beating relation b^p if $p_x \preceq(b^p)\, p_y$.

$$p \preceq(b^p)\, p \tag{REFL}$$

$$p_x \preceq(b^p)\, p_y \wedge p_y \preceq(b^p)\, p_z \Rightarrow p_x \preceq(b^p)\, p_z \tag{TRAN}$$

$$p_x \not\preceq(b^p)\, p_y \Rightarrow p_x \preceq(b^p)\, p_y \tag{COMP}$$

4.2.4 The Algebraic Structure of Beating Functions

The set B^P and pointwise natural addition operation $(b^p_x + b^p_y)(p_w, p_l, s_{wc}, s_{lc}, s_w) = b^p_x(p_w, p_l, s_{wc}, s_{lc}, s_w) + b^p_y(p_w, p_l, s_{wc}, s_{lc}, s_w)$ form an algebra. The pointwise natural addition operation is associative, commutative and $\mathbf{b^p_0}$ is its identity element. $\mathbf{b^p_0}$

is the beating function representing the results of the empty set of semantic games. Therefore, $\mathbf{b_0^p}(p_w, p_l, s_{wc}, s_{lc}, s_w) = 0$.

We add the following four restriction operations to the algebra:

- **Win restriction**: we use $b^p|^w_{p_x}$ to denote a restricted version of b^p that only contains those games that p_x wins. Formally,

$$b^p|^w_{p_x}(p_w, p_l, s_{wc}, s_{lc}, s_w) = \begin{cases} b^p(p_w, p_l, s_{wc}, s_{lc}, s_w) & , p_w = p_x \\ 0 & , otherwise \end{cases} \tag{DEF.WR}$$

- **Loss restriction**: we use $b^p|^l_{p_x}$ to denote a restricted version of b^p that only contains those games that p_x loses. Formally,

$$b^p|^l_{p_x}(p_w, p_l, s_{wc}, s_{lc}, s_w) = \begin{cases} b^p(p_w, p_l, s_{wc}, s_{lc}, s_w) & , p_l = p_x \\ 0 & , otherwise \end{cases} \tag{DEF.LR}$$

- **Fault restriction**: we use $b^p|^{fl}_{p_x}$ to denote a restricted version of b^p that only contains the games in which p_x makes a fault. These are the games that p_x had a chance to win yet it lost. Formally,

$$b^p|^{fl}_{p_x}(p_w, p_l, s_{wc}, s_{lc}, s_w) = \begin{cases} b^p(p_w, p_l, s_{wc}, s_{lc}, s_w) & , p_l = p_x \wedge s_{lc} \neq s_w \\ 0 & , otherwise \end{cases} \tag{DEF.FR}$$

- **Control restriction**: we use $b^p|^c_{p_x}$ to denote a restricted version of b^p that only contains those games that p_x controls. These are the games that p_x either wins or had a chance to win. Formally:

$$b^p|^c_{p_x} = b^p|^w_{p_x} + b^p|^{fl}_{p_x} \tag{DEF.CR}$$

We also add a complement restriction operation for each of the aforementioned restriction operations. We use $b^p|^{!w}_{p_x}$ to denote a restricted version of b^p that only contains those games that p_x does not win. Formally:

$$b^p|^{w}_{p_x} + b^p|^{!w}_{p_x} = b^p \tag{DEF.CWR}$$

We use $b^p|^{!l}_{p_x}$ to denote a restricted version of b^p that only contains those games that p_x does not lose. Formally:

$$b^p|^{l}_{p_x} + b^p|^{!l}_{p_x} = b^p \tag{DEF.CLR}$$

We use $b^p|^{!fl}_{p_x}$ to denote a restricted version of b^p that only contains the games in which p_x does not make a fault. Formally:

$$b^p|^{fl}_{p_x} + b^p|^{!fl}_{p_x} = b^p \tag{DEF.CFR}$$

We use $b^p|^{!c}_{p_x}$ to denote a restricted version of b^p that only contains those games that p_x does not control. Formally:

$$b^p|^{c}_{p_x} + b^p|^{!c}_{p_x} = b^p \tag{DEF.CCR}$$

Formally:

$$b^p|^{fl}_{p_y}|^{fl}_{p_x} = \mathbf{b^p_0} \tag{DBL.R.F}$$

$$b^p|^{w}_{p_y}|^{w}_{p_x} = \mathbf{b^p_0} \tag{DBL.R.W}$$

$$b^p|^{!fl}_{p_x} = b^p|^{w}_{p_x} + b^p|^{!c}_{p_x} \tag{PROP.I}$$

$$b^p|^{!c}_{p_x}|^{c}_{p_x} = \mathbf{b^p_0} \tag{PROP.II}$$

$$b^p|^{fl}_{p_x}|^{l}_{p_x} = b^p|^{fl}_{p_x} \tag{PROP.III}$$

4.3 Collusion Resilient Ranking Functions

In this section, we formalize our notion of collusion resilient ranking functions as ranking functions satisfying the limited collusion effect property. Then we formalize the other desirable property of never discouraging wins nor encouraging losses. Finally, we provide a more practical alternative characterization of collusion resilient ranking functions that that never discourage winning nor encourage losing.

4.3.1 Limited Collusion Effect

A ranking function $\preceq_r$ is said to have the **Limited Collusion Effect (LCE)** property if for any two arbitrary players p_x and p_y the rank of p_y with respect to p_x cannot be improved by manipulating games that p_x can not control their outcome. These are the games that p_x is not involved in or the games p_x loses while forced. Formally, a ranking function satisfies the LCE property if it satisfies the following axioms:

$$b_2^p|_{p_x}^c = \mathbf{b_0^p} \wedge p_x \preceq(b_1^p)\, p_y \Rightarrow p_x \preceq(b_1^p + b_2^p)\, p_y \qquad \text{(LCE.I)}$$

$$b_2^p|_{p_x}^c = \mathbf{b_0^p} \wedge p_y \npreceq(b_1^p)\, p_x \Rightarrow p_y \npreceq(b_1^p + b_2^p)\, p_x \qquad \text{(LCE.II)}$$

The first axiom asserts that if p_x is ranked weakly better p_y under the beating function b_1^p, then p_x remains weakly better than p_y when more games that p_x cannot control are added to b_1^p. The second axiom asserts that if p_x is ranked strictly better p_y under the beating function b_1^p, then p_x remains strictly better than p_y when more games that p_x cannot control are added to b_1^p.

4.3.2 Regard for Wins and Losses

It is unacceptable for a ranking function to reward losing or to penalize winning. In other words, a ranking function must have a **Non-Negative Regard for Winning (NNRW)** and a **Non-Positive Regard for Losing (NPRL)**. That is, a player's rank cannot be worsened by an extra winning nor can it be improved by an extra loss. Formally, a ranking function must satisfy the following axioms:

$$p_x \preceq(b_1^p)\, p_y \Rightarrow p_x \preceq(b_1^p + b_2^p|_{p_x}^{w})\, p_y \tag{NNRW.I}$$

$$p_x \preceq(b_1^p + b_2^p|_{p_y}^{w})\, p_y \Rightarrow p_x \preceq(b_1^p)\, p_y \tag{NNRW.II}$$

$$p_x \preceq(b_1^p)\, p_y \Rightarrow p_x \preceq(b_1^p + b_2^p|_{p_y}^{l})\, p_y \tag{NPRL.I}$$

$$p_x \preceq(b_1^p + b_2^p|_{p_x}^{l})\, p_y \Rightarrow p_x \preceq(b_1^p)\, p_y \tag{NPRL.II}$$

4.3.3 A Practical Characterization of Collusion Resilient Ranking Functions

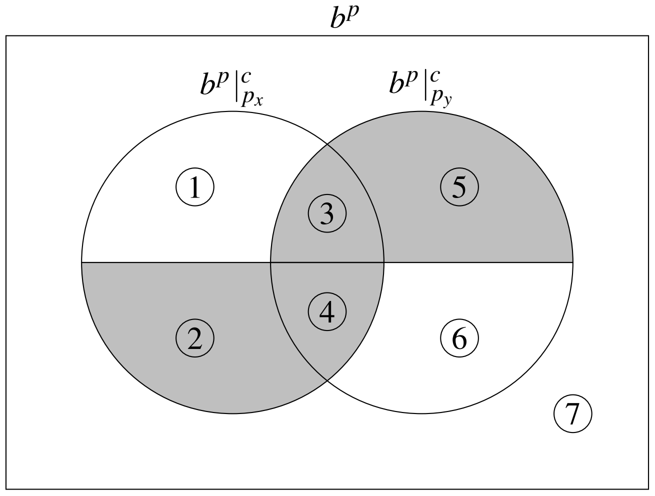
We now show that: in the context of ranking functions satisfying both NNRW and NPRL axioms, there is an equivalent, yet a more practical, alternative characterization of collusion resilient ranking functions. We start by formalizing the alternative axiom. Then we formally prove a theorem formalizing the relationship between the two characterizations.

A ranking function $\preceq$ is said to be **Local Fault Based (LFB)** if for any two arbitrary players p_x and p_y the relative rank $\preceq$ assigns to p_x with respect to p_y solely depends on the games where p_x or p_y make a fault. Formally,

$$p_x \preceq(b^p|_{p_x}^{fl} + b^p|_{p_y}^{fl})\, p_y \Leftrightarrow p_x \preceq(b^p)\, p_y \tag{LFB}$$

Theorem 4.3.1. *For any ranking function having NNRW and NPRL, LCE is equivalent to LFB. Formally, NNRW $\wedge$ NPRL $\Rightarrow$ (LCE $\Leftrightarrow$ LFB).*

Figure 4.1 presents a partitioning of the games represented by an arbitrary beating function b^p. This partitioning illustrates the intuition behind this theorem and its proof. The intuition is that a ranking function satisfying the LFB property



① $b^p|^{!fl}_{p_y}|^{w}_{p_x} = b^p|^{!c}_{p_y}|^{w}_{p_x}$
② $b^p|^{!w}_{p_y}|^{fl}_{p_x} = b^p|^{!c}_{p_y}|^{fl}_{p_x}$
③ $b^p|^{fl}_{p_y}|^{w}_{p_x} = b^p|^{c}_{p_y}|^{w}_{p_x}$
④ $b^p|^{w}_{p_y}|^{fl}_{p_x} = b^p|^{c}_{p_y}|^{fl}_{p_x}$
⑤ $b^p|^{!w}_{p_x}|^{fl}_{p_y} = b^p|^{!c}_{p_x}|^{fl}_{p_y}$
⑥ $b^p|^{!fl}_{p_x}|^{w}_{p_y} = b^p|^{!c}_{p_x}|^{w}_{p_y}$
⑦ $b^p|^{!c}_{p_x}|^{!c}_{p_y}$

Figure 4.1: Beating Functions Representing Partitions of the Semantic Games Represented by b^p

$\preceq$ must completely decide the relative rank of any two arbitrary players p_x and p_y based on the games in the shaded partitions only. Games in the unshaded partitions cannot influence the relative rank of p_x and p_y assigned by $\preceq$.

We now give an informal proof of this theorem using the partitioning shown in Figure 4.1. We break our theorem into the following two lemmas. The first lemma is that NNRW and LCE imply LFB. The second lemma is that NPRL and LFB imply LCE. Our theorem follows directly from both lemmas.

To prove the first lemma, let $\preceq$ be a ranking function that violates the LFB property. By definition of the LFB property, there must be two players p_x and p_y such that the games in the unshaded region influence the relative rank assigned by $\preceq$ to p_x and p_y. The influence can either be positive (case I) or negative (case II) for p_x. Suppose that games in the unshaded region positively influence the rank assigned by $\preceq$ to p_x with respect to p_y. But, assuming that $\preceq$ satisfies the LCE property, games in partitions ①, ⑦ cannot improve p_x's rank with respect to p_y because it only contains games not under p_y's control. Also, assuming that $\preceq$ satisfies the NNRW property, games in partition ⑥ cannot improve p_y's rank with respect to p_x because it only contains games that p_y has won. Therefore, our assumption that $\preceq$ satisfies both LCE and NNRW cannot be true. We have shown the contrapositive of the first lemma for case I. We now consider case II. Suppose that games in the unshaded region negatively influence the rank assigned by $\preceq$ to p_x with respect to p_y. But assuming that $\preceq$ satisfies the LCE property, games in partitions ⑥, ⑦ cannot worsen p_x's rank with respect to p_y because it only contains games not under p_x's control. Also, assuming that $\preceq$ satisfies the NNRW property, games in partition ① cannot worsen p_x's rank with respect to p_y because it only contains games that p_x has won. Therefore, our assumption that $\preceq$ satisfies both LCE and NNRW cannot be true. We have shown the contrapositive of the first lemma for case II and the first part of the proof is now complete.

To prove the second lemma, let $\preceq$ be a ranking function satisfying both NPRL and LFB. By definition of the LFB property, only games in the shaded region influence the relative rank assigned by $\preceq$ to p_x and p_y. Games in the regions ②,③ and ④ are under the control of p_x. Only games in region ⑤ can influence the relative rank assigned by $\preceq$ to p_x and p_y, yet games in region ⑤ are not under the control of p_x. However, games in region ⑤ are all faults made by p_y and by NPRL they cannot improve the rank of p_y with respect to p_x. Therefore, only games under the control of p_x may worsen p_x's rank with respect to p_y. An identical argument applies to the rank of p_y. This completes the prove of the second lemma and hence the theorem.

Now we present our formal proof. We start with few lemmas. The first lemma formalizes the partitioning shown in Figure 4.1. Essentially, our first lemma asserts that if we add the games in the shaded region, represented by the beating function $b^p|^{fl}_{p_x} + b^p|^{fl}_{p_y}$, to the games in partitions ①, ⑥ and ⑦ represented by the beating functions $b^p|^{!fl}_{p_y}|^{w}_{p_x}$, $b^p|^{!fl}_{p_x}|^{w}_{p_y}$ and $b^p|^{!c}_{p_x}|^{!c}_{p_y}$ respectively, we get b^p.

Lemma 4.3.2. $b^p|^{fl}_{p_x} + b^p|^{fl}_{p_y} + b^p|^{!fl}_{p_y}|^{w}_{p_x} + b^p|^{!fl}_{p_x}|^{w}_{p_y} + b^p|^{!c}_{p_x}|^{!c}_{p_y} = b^p$

Proof.

$b^p|^{fl}_{p_x} + b^p|^{fl}_{p_y} + b^p|^{!fl}_{p_y}|^{w}_{p_x} + b^p|^{!fl}_{p_x}|^{w}_{p_y} + b^p|^{!c}_{p_y}|^{!c}_{p_x}$

By DEF.CCR$[b^p|^{fl}_{p_y}/b^p]$:

$=b^p|^{fl}_{p_x} + b^p|^{fl}_{p_y}|^{c}_{p_x} + b^p|^{fl}_{p_y}|^{!c}_{p_x} + b^p|^{!fl}_{p_y}|^{w}_{p_x} + b^p|^{!fl}_{p_x}|^{w}_{p_y} + b^p|^{!c}_{p_y}|^{!c}_{p_x}$

By DEF.CR$[b^p|^{fl}_{p_y}/b^p]$:

$=b^p|^{fl}_{p_x} + b^p|^{fl}_{p_y}|^{w}_{p_x} + b^p|^{fl}_{p_y}|^{fl}_{p_x} + b^p|^{fl}_{p_y}|^{!c}_{p_x} + b^p|^{!fl}_{p_y}|^{w}_{p_x} + b^p|^{!fl}_{p_x}|^{w}_{p_y} + b^p|^{!c}_{p_y}|^{!c}_{p_x}$

By DBL.R.F and identity of + :

$=b^p|^{fl}_{p_x} + b^p|^{fl}_{p_y}|^{w}_{p_x} + b^p|^{fl}_{p_y}|^{!c}_{p_x} + b^p|^{!fl}_{p_y}|^{w}_{p_x} + b^p|^{!fl}_{p_x}|^{w}_{p_y} + b^p|^{!c}_{p_y}|^{!c}_{p_x}$

By commutativity of + and distributivity of + on restrictions :

$=b^p|^{fl}_{p_x} + (b^p|^{fl}_{p_y} + b^p|^{!fl}_{p_y})|^{w}_{p_x} + b^p|^{fl}_{p_y}|^{!c}_{p_x} + b^p|^{!fl}_{p_x}|^{w}_{p_y} + b^p|^{!c}_{p_y}|^{!c}_{p_x}$

By DEF.CFR :

$=b^p|^{fl}_{p_x} + b^p|^{w}_{p_x} + b^p|^{fl}_{p_y}|^{!c}_{p_x} + b^p|^{!fl}_{p_x}|^{w}_{p_y} + b^p|^{!c}_{p_y}|^{!c}_{p_x}$

By commutativity of + and DEF.CR :

$=b^p|^{c}_{p_x} + b^p|^{fl}_{p_y}|^{!c}_{p_x} + b^p|^{!fl}_{p_x}|^{w}_{p_y} + b^p|^{!c}_{p_y}|^{!c}_{p_x}$

By PROP.I :

$=b^p|^{c}_{p_x} + b^p|^{fl}_{p_y}|^{!c}_{p_x} + (b^p|^{w}_{p_x} + b^p|^{!c}_{p_x})|^{w}_{p_y} + b^p|^{!c}_{p_y}|^{!c}_{p_x}$

By distributivity of + on restriction operations:

$=b^p|^{c}_{p_x} + b^p|^{fl}_{p_y}|^{!c}_{p_x} + b^p|^{w}_{p_x}|^{w}_{p_y} + b^p|^{!c}_{p_x}|^{w}_{p_y} + b^p|^{!c}_{p_y}|^{!c}_{p_x}$

By DBL.R.W and identity of + :

$=b^p|^{c}_{p_x} + b^p|^{fl}_{p_y}|^{!c}_{p_x} + b^p|^{!c}_{p_x}|^{w}_{p_y} + b^p|^{!c}_{p_y}|^{!c}_{p_x}$

By commutativity of + and restrictions and distributivity of + on restrictions :

$=b^p|^{c}_{p_x} + (b^p|^{w}_{p_y} + b^p|^{fl}_{p_y})|^{!c}_{p_x} + b^p|^{!c}_{p_y}|^{!c}_{p_x}$

By DEF.CR$[p_y/p_x]$:

$=b^p|^{c}_{p_x}+b^p|^{c}_{p_y}|^{!c}_{p_x}+b^p|^{!c}_{p_y}|^{!c}_{p_x}$

By distributivity of + on restrictions :

$=b^p|^{c}_{p_x}+(b^p|^{c}_{p_y}+b^p|^{!c}_{p_y})|^{!c}_{p_x}$

By DEF.CCR$[p_y/p_x]$ and commutativity of + :

$=b^p|^{c}_{p_x}+b^p|^{!c}_{p_x}$

By DEF.CCR and commutativity of + :

$=b^p$

$\square$

Our second lemma asserts that partitions ① and ⑦, represented by the beating functions $b^p|^{!fl}_{p_y}|^{w}_{p_x}$ and $b^p|^{!c}_{p_x}|^{!c}_{p_y}$ respectively, are not under p_y's control.

Lemma 4.3.3.

$$(b^p|^{!fl}_{p_y}|^{w}_{p_x}+b^p|^{!c}_{p_x}|^{!c}_{p_y})|^{c}_{p_y}=\mathbf{b^p_0}$$

Proof.

$(b^p|^{!fl}_{p_y}|^{w}_{p_x}+b^p|^{!c}_{p_x}|^{!c}_{p_y})|^{c}_{p_y}$

By PROP.I$[p_y/p_x]$:

$=(b^p|^{w}_{p_y}|^{w}_{p_x}+b^p|^{!c}_{p_y}|^{w}_{p_x}+b^p|^{!c}_{p_x}|^{!c}_{p_y})|^{c}_{p_y}$

By DBL.R.W and identity of + :

$=(b^p|^{!c}_{p_y}|^{w}_{p_x}+b^p|^{!c}_{p_x}|^{!c}_{p_y})|^{c}_{p_y}$

By commutativity of + and restrictions and distributivity of + on restrictions :

$=((b^p|^{w}_{p_x}+b^p|^{!c}_{p_x})|^{!c}_{p_y})|^{c}_{p_y}$

By PROP.II$[p_y/p_x,(b^p|^{w}_{p_x}+b^p|^{!c}_{p_x})/b^p]$:

$=\mathbf{b^p_0}$

$\square$

Our third lemma asserts that NNRW and LCE imply LFB.

Lemma 4.3.4. *NNRW $\wedge$ LCE $\Rightarrow$ LFB.*

Proof. We show the contrapositive of the lemma; Let $\preceq$ be a ranking function violating LFB. Formally, $p_x \preceq (b^p|^{fl}_{p_x} + b^p|^{fl}_{p_y})\, p_y \not\Leftrightarrow p_x \preceq (b^p)\, p_y$. We show that $(p_x \preceq (b^p|^{fl}_{p_x} + b^p|^{fl}_{p_y})\, p_y \not\Leftrightarrow p_x \preceq (b^p)\, p_y) \Rightarrow$ **false** under the assumption that $\preceq$ satisfies both LCE and NNRW properties.

$p_x \preceq (b^p|^{fl}_{p_x} + b^p|^{fl}_{p_y})\, p_y \not\Leftrightarrow p_x \preceq (b^p)\, p_y$

$\Rightarrow p_x \npreceq (b^p|^{fl}_{p_x} + b^p|^{fl}_{p_y})\, p_y \wedge p_x \preceq (b^p)\, p_y \vee p_x \preceq (b^p|^{fl}_{p_x} + b^p|^{fl}_{p_y})\, p_y \wedge p_x \npreceq (b^p)\, p_y$ (I)

Consider the left disjunct only :

$p_x \npreceq (b^p|^{fl}_{p_x} + b^p|^{fl}_{p_y})\, p_y \wedge p_x \preceq (b^p)\, p_y$

Using Lemma 4.3.3 :

$\Rightarrow ((b^p|^{!fl}_{p_y}|^{w}_{p_x} + b^p|^{!c}_{p_x}|^{!c}_{p_y})|^{c}_{p_y} = \mathbf{b^p_0}) \wedge p_x \npreceq (b^p|^{fl}_{p_x} + b^p|^{fl}_{p_y})\, p_y \wedge p_x \preceq (b^p)\, p_y$

By LCE.II$[p_y/p_x, p_x/p_y, (b^p|^{fl}_{p_x} + b^p|^{fl}_{p_y})/b^p_1, (b^p|^{!fl}_{p_y}|^{w}_{p_x} + b^p|^{!c}_{p_x}|^{!c}_{p_y})/b^p_2]$:

$\Rightarrow p_x \npreceq (b^p|^{fl}_{p_x} + b^p|^{fl}_{p_y} + b^p|^{!fl}_{p_y}|^{w}_{p_x} + b^p|^{!c}_{p_x}|^{!c}_{p_y})\, p_y \wedge p_x \preceq (b^p)\, p_y$

By the contrapositive of NNRW.II$[(b^p|^{fl}_{p_x} + b^p|^{fl}_{p_y} + b^p|^{!fl}_{p_y}|^{w}_{p_x} + b^p|^{!c}_{p_x}|^{!c}_{p_y})/b^p_1, b^p|^{!fl}_{p_x}/b^p_2]$:

$\Rightarrow p_x \npreceq (b^p|^{fl}_{p_x} + b^p|^{fl}_{p_y} + b^p|^{!fl}_{p_y}|^{w}_{p_x} + b^p|^{!c}_{p_x}|^{!c}_{p_y} + b^p|^{!fl}_{p_x}|^{w}_{p_y})\, p_y \wedge p_x \preceq (b^p)\, p_y$

By Lemma 4.3.2 and commutativity of + and restrictions :

$\Rightarrow p_x \npreceq (b^p)\, p_y \wedge p_x \preceq (b^p)\, p_y$

$\Rightarrow$**false** (II)

Consider the right disjunct only :

$p_x \preceq (b^p|^{fl}_{p_x} + b^p|^{fl}_{p_y})\, p_y \wedge p_x \npreceq (b^p)\, p_y$

Using Lemma 4.3.3$[p_y/p_x, p_x/p_y]$:

$\Rightarrow((b^p|^{!fl}_{p_x}|^{w}_{p_y} + b^p|^{!c}_{p_y}|^{!c}_{p_x})|^{c}_{p_x} = \mathbf{b^p_0}) \wedge p_x \preceq(b^p|^{fl}_{p_x} + b^p|^{fl}_{p_y})\, p_y \wedge p_x \npreceq(b^p)\, p_y$

By LCE.I$[(b^p|^{fl}_{p_x} + b^p|^{fl}_{p_y})/b^p_1, (b^p|^{!fl}_{p_x}|^{w}_{p_y} + b^p|^{!c}_{p_y}|^{!c}_{p_x})/b^p_2]$:

$\Rightarrow p_x \preceq(b^p|^{fl}_{p_x} + b^p|^{fl}_{p_y} + b^p|^{!fl}_{p_x}|^{w}_{p_y} + b^p|^{!c}_{p_y}|^{!c}_{p_x})\, p_y \wedge p_x \npreceq(b^p)\, p_y$

By NNRW.I$[(b^p|^{fl}_{p_x} + b^p|^{fl}_{p_y} + b^p|^{!fl}_{p_x}|^{w}_{p_y} + b^p|^{!c}_{p_y}|^{!c}_{p_x})/b^p_1, b^p|^{!fl}_{p_y}/b^p_2]$:

$\Rightarrow p_x \preceq(b^p|^{fl}_{p_x} + b^p|^{fl}_{p_y} + b^p|^{!fl}_{p_x}|^{w}_{p_y} + b^p|^{!c}_{p_y}|^{!c}_{p_x} + b^p|^{!fl}_{p_y}|^{w}_{p_x})\, p_y \wedge p_x \npreceq(b^p)\, p_y$

By Lemma 4.3.2 and commutativity of + and restrictions :

$\Rightarrow p_x \preceq(b^p)\, p_y \wedge p_x \npreceq(b^p)\, p_y$

$\Rightarrow$**false** (III)

From I , II and II :

$p_x \preceq(b^p|^{fl}_{p_x} + b^p|^{fl}_{p_y})\, p_y \not\Leftrightarrow p_x \preceq(b^p)\, p_y) \Rightarrow$ **false**

□

Our forth lemma asserts that NPRL and LFB imply LCE.

Lemma 4.3.5. *NPRL $\wedge$ LFB $\Rightarrow$ LCE.*

Proof. We separately derive the R.H.S. of each of the LCE axioms from its corresponding L.H.S. under the assumptions of NPRL and LFB.

Consider the L.H.S. of LCE.I:

$b^p_2|^{c}_{p_x} = \mathbf{b^p_0} \wedge p_x \preceq(b^p_1)\, p_y$

Using LFB:

$\Rightarrow b^p_2|^{c}_{p_x} = \mathbf{b^p_0} \wedge p_x \preceq(b^p_1|^{fl}_{p_x} + b^p_1|^{fl}_{p_y})\, p_y$

By DEF.CR:

$\Rightarrow b^p_2|^{fl}_{p_x} + b^p_2|^{w}_{p_x} = \mathbf{b^p_0} \wedge p_x \preceq(b^p_1|^{fl}_{p_x} + b^p_1|^{fl}_{p_y})\, p_y$

By definition of a beating function, it must return a natural number :

$\Rightarrow b_2^p|_{p_x}^{fl} = \mathbf{b_0^p} \wedge p_x \preceq (b_1^p|_{p_x}^{fl} + b_1^p|_{p_y}^{fl})\, p_y$

Since $\mathbf{b_0^p}$ is the identity element for + :

$\Rightarrow p_x \preceq (b_1^p|_{p_x}^{fl} + b_2^p|_{p_x}^{fl} + b_1^p|_{p_y}^{fl})\, p_y$

By NPRL.I$[b_2^p|_{p_y}^{fl}/b_2^p]$:

$\Rightarrow p_x \preceq (b_1^p|_{p_x}^{fl} + b_2^p|_{p_x}^{fl} + b_1^p|_{p_y}^{fl} + b_2^p|_{p_y}^{fl}|_{p_y}^{l})\, p_y$

By PROP.III$[p_y/p_x, b_2^p/b^p]$:

$\Rightarrow p_x \preceq (b_1^p|_{p_x}^{fl} + b_2^p|_{p_x}^{fl} + b_1^p|_{p_y}^{fl} + b_2^p|_{p_y}^{fl})\, p_y$

By distributivity:

$\Rightarrow p_x \preceq (b_1^p + b_2^p|_{p_x}^{fl} + b_1^p + b_2^p|_{p_y}^{fl})\, p_y$

By LFB:

$\Rightarrow p_x \preceq (b_1^p + b_2^p)\, p_y = R.H.S.$ (I)

Consider the L.H.S. of LCE.II:

$b_2^p|_{p_x}^{c} = \mathbf{b_0^p} \wedge p_y \npreceq (b_1^p)\, p_x$

Using LFB:

$\Rightarrow b_2^p|_{p_x}^{c} = \mathbf{b_0^p} \wedge p_y \npreceq (b_1^p|_{p_x}^{fl} + b_1^p|_{p_y}^{fl})\, p_x$

By DEF.CR:

$\Rightarrow b_2^p|_{p_x}^{fl} + b_2^p|_{p_x}^{w} = \mathbf{b_0^p} \wedge p_y \npreceq (b_1^p|_{p_x}^{fl} + b_1^p|_{p_y}^{fl})\, p_x$

By definition of a beating function, it must return a natural number :

$\Rightarrow b_2^p|_{p_x}^{fl} = \mathbf{b_0^p} \wedge p_y \npreceq (b_1^p|_{p_x}^{fl} + b_1^p|_{p_y}^{fl})\, p_x$

Since $\mathbf{b_0^p}$ is the identity element for + :

$\Rightarrow p_y \npreceq (b_1^p|_{p_x}^{fl} + b_2^p|_{p_x}^{fl} + b_1^p|_{p_y}^{fl})\, p_x$

By the contrapositive of NPRL.II$[p_y/p_x, b_2^p|_{p_y}^{fl}/b_2^p]$:

$\Rightarrow p_y \not\preceq (b_1^p|_{p_x}^{fl} + b_2^p|_{p_x}^{fl} + b_1^p|_{p_y}^{fl} + b_2^p|_{p_y}^{fl}|_{p_y}^{l})\, p_x$

By PROP.III$[p_y/p_x, b_2^p/b^p]$:

$\Rightarrow p_y \not\preceq (b_1^p|_{p_x}^{fl} + b_2^p|_{p_x}^{fl} + b_1^p|_{p_y}^{fl} + b_2^p|_{p_y}^{fl})\, p_x$

By distributivity:

$\Rightarrow p_y \not\preceq (b_1^p + b_2^p|_{p_x}^{fl} + b_1^p + b_2^p|_{p_y}^{fl})\, p_x$

By LFB:

$\Rightarrow p_y \not\preceq (b_1^p + b_2^p)\, p_x = R.H.S.$ (II)

From I, II:

NPRL $\wedge$ LFB $\Rightarrow$ LCE

□

We now proceed to prove Theorem 4.3.1.

Proof.

By 4.3.4:

NNRW $\wedge$ LCE $\Rightarrow$ LFB

Therefore:

NNRW $\wedge$ NPRL $\wedge$ LCE $\Rightarrow$ LFB (I)

By 4.3.5

NPRL $\wedge$ LFB $\Rightarrow$ LCE

Therefore:

NNRW $\wedge$ NPRL $\wedge$ LFB $\Rightarrow$ LCE (II)

Combining I $\wedge$ II :

$(\text{NNRW} \wedge \text{NPRL} \wedge \text{LCE} \Rightarrow \text{LFB}) \wedge (\text{NNRW} \wedge \text{NPRL} \wedge \text{LFB} \Rightarrow \text{LCE})$

Simplifying:

$\text{NNRW} \wedge \text{NPRL} \Rightarrow (\text{LCE} \Leftrightarrow \text{LFB})$

$\square$

4.3.4 Neutrality

It is unacceptable that a ranking function to rank players based on their identity. Instead, a ranking function must solely depend on the structure of the input beating function. We call this property **NEUtrality (NEU)**. A ranking function $\preceq$ is said to be neutral if the order $\preceq(b^p)$ it assigns to some beating function b^p is preserved under any permutation on P preserving b^p. We use a^{b^p} to denote a permutation on P that preserves b^p. Formally, $b^p(p_w, p_l, s_{wc}, s_{lc}, s_w) = b^p(a^{b^p}(p_w), a^{b^p}(p_l), s_{wc}, s_{lc}, s_w)$. We formally define Neutrality as:

$$p_x \preceq(b^p)\, p_y \Leftrightarrow a^{b^p}(p_x) \preceq(b^p)\, a^{b^p}(p_y) \tag{NEU}$$

4.4 Characterization of the Fault Counting Ranking Function

Definition 4.4.1. *Let $faults^{b^p}(p)$ be the number of faults that player p makes in b^p. Formally:*

$$faults^{b^p}(p) = \sum_{p_w, p_l \in P \wedge s_{wc}, s_{lc}, s_w \in \mathbf{S}} b^p|_p^{fl}(p_w, p_l, s_{wc}, s_{lc}, s_w) \tag{DEF.FLTS}$$

***Fault counting** is a ranking function in which players are ranked according to the number of faults they incur; The fewer the number of faults the better the rank. Formally:*

$$p_x \preceq_f(b^p)\, p_y = faults^{b^p}(p_x) \leq faults^{b^p}(p_y) \tag{DEF.FC}$$

4.4.1 Neutrality

A ranking function $\preceq$ is said to be **neutral** if the order $\preceq(b^p)$ it assigns to some beating function b^p is preserved under any permutation on P preserving b^p. We use a^{b^p} to denote a permutation on P that preserves b^p. Formally, $b^p(p_w, p_l, s_{wc}, s_{lc}, s_w) = b^p(a^{b^p}(p_w), a^{b^p}(p_l), s_{wc}, s_{lc}, s_w)$. We formally define Neutrality as:

$$p_x \preceq(b^p)\, p_y \Leftrightarrow a^{b^p}(p_x) \preceq(b^p)\, a^{b^p}(p_y) \tag{NEU}$$

[Why is neutrality important?]

4.4.2 Most Refined

Let $\preceq_r$, $\preceq_s$ be two ranking functions. We say that $\preceq_r$ **refines** $\preceq_s$ if $p_x \preceq_r(b^p)\, p_y \Rightarrow p_x \preceq_s(b^p)\, p_y$. Intuitively, the equivalence classes of $\sim^r$ can be produced via partitioning some of the equivalence classes of $\sim^s$.

[Why is refinement a desirable property?]

The following theorem provides a complete characterization of the loss counting ranking function.

Theorem 4.4.2. *$\preceq_f$ is the most refined, neutral ranking function that satisfies the limited collusion effect property and has a positive regard for wins.*

Lemma 4.4.3. *$\preceq_f$ satisfies the LCE property. Formally:*

$$(b_2^p|^c_{p_x} = \mathbf{b_0^p} \wedge p_x \preceq_f(b_1^p)\, p_y \Rightarrow p_x \preceq_f(b_1^p + b_2^p)\, p_y) \wedge$$

$$(b_2^p|^c_{p_x} = \mathbf{b_0^p} \wedge p_y \not\preceq_f(b_1^p)\, p_x \Rightarrow p_y \not\preceq_f(b_1^p + b_2^p)\, p_x)$$

Proof.

Consider the left conjunct of both clauses :

$b_2^p|_{p_x}^c = \mathbf{b_0^p}$

Since $faults()$ is a function :

$\Rightarrow faults^{b_2^p|_{p_x}^c}(p_x) = faults^{\mathbf{b_0^p}}(p_x)$

By DEF.FLTS :

$$\Rightarrow \sum_{p_w,p_l \in P \wedge s_{wc},s_{lc},s_w \in \mathbf{S}} b_2^p|_{p_x}^c|_{p_x}^{fl}(p_w,p_l,s_{wc},s_{lc},s_w) = \mathbf{0}$$

By def of control restrict and distributivity :

$$\Rightarrow \sum_{p_w,p_l \in P \wedge s_{wc},s_{lc},s_w \in \mathbf{S}} (b_2^p|_{p_x}^w|_{p_x}^{fl} + b_2^p|_{p_x}^{fl}|_{p_x}^{fl})(p_w,p_l,s_{wc},s_{lc},s_w) = \mathbf{0}$$

By $b^p|_{p_x}^w|_{p_x}^{fl} = \mathbf{b_0^p}$ and identity :

$$\Rightarrow \sum_{p_w,p_l \in P \wedge s_{wc},s_{lc},s_w \in \mathbf{S}} b_2^p|_{p_x}^{fl}|_{p_x}^{fl}(p_w,p_l,s_{wc},s_{lc},s_w) = \mathbf{0}$$

By idempotence of $b^p|_{p_x}^{fl}$:

$$\Rightarrow \sum_{p_w,p_l \in P \wedge s_{wc},s_{lc},s_w \in \mathbf{S}} b_2^p|_{p_x}^{fl}(p_w,p_l,s_{wc},s_{lc},s_w) = \mathbf{0}$$

By DEF.FLTS :

$$\Rightarrow faults^{b_2^p}(p_x) = \mathbf{0} \tag{I}$$

Now consider the right conjunct of the first clause :

L.H.S. $= p_x \preceq_f(b_1^p)\, p_y$

$\Rightarrow faults^{b_1^p}(p_x) \leq faults^{b_1^p}(p_y)$

$\Rightarrow faults^{b_1^p}(p_x) \leq faults^{b_1^p}(p_y) + faults^{b_2^p}(p_y)$

$\Rightarrow faults^{b_1^p}(p_x) + faults^{b_2^p}(p_x) \leq faults^{b_1^p}(p_y) + faults^{b_2^p}(p_y)$

$\Rightarrow faults^{b_1^p+b_2^p}(p_x) \leq faults^{b_1^p+b_2^p}(p_y)$

$$\Rightarrow p_x \preceq_f(b_1^p + b_2^p)\, p_y = \text{R.H.S.} \tag{II}$$

Now consider the right conjunct of the second clause :

L.H.S. $= p_y \not\preceq_f(b_1^p)\, p_x$

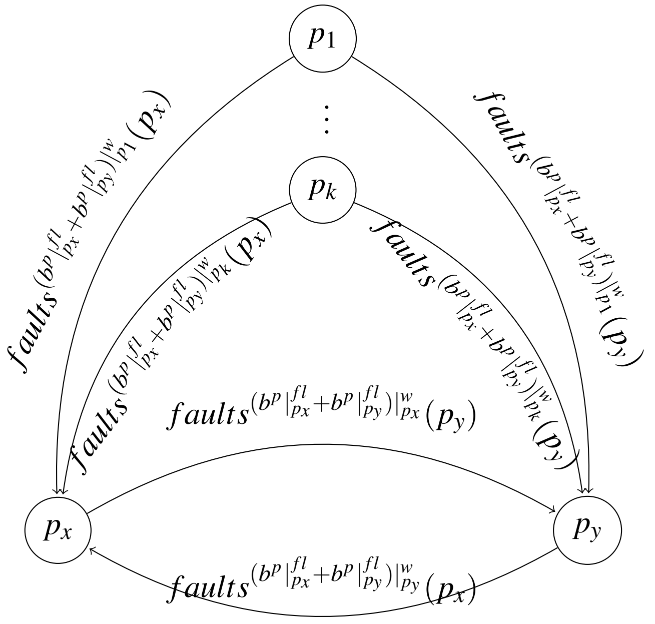

p_1
p_k
p_x
p_y
$faults^{(b^p|^{fl}_{p_x}+b^p|^{fl}_{p_y})|^w_{p_1}}(p_x)$
$faults^{(b^p|^{fl}_{p_x}+b^p|^{fl}_{p_y})|^w_{p_1}}(p_y)$
$faults^{(b^p|^{fl}_{p_x}+b^p|^{fl}_{p_y})|^w_{p_k}}(p_x)$
$faults^{(b^p|^{fl}_{p_x}+b^p|^{fl}_{p_y})|^w_{p_k}}(p_y)$
$faults^{(b^p|^{fl}_{p_x}+b^p|^{fl}_{p_y})|^w_{p_x}}(p_y)$
$faults^{(b^p|^{fl}_{p_x}+b^p|^{fl}_{p_y})|^w_{p_y}}(p_x)$

Figure 4.2: Partitioning faults made by either p_x, p_y or both

$$\Rightarrow faults^{b_1^p}(p_y) \npreceq faults^{b_1^p}(p_x)$$

$$\Rightarrow faults^{b_1^p}(p_y) + faults^{b_2^p}(p_y) \npreceq faults^{b_1^p}(p_x) + faults^{b_2^p}(p_x)$$

$$\Rightarrow faults^{b_1^p+b_2^p}(p_y) \npreceq faults^{b_1^p+b_2^p}(p_x)$$

$$\Rightarrow p_y \npreceq_f (b_1^p + b_2^p)\, p_x = \text{R.H.S.} \qquad \text{(III)}$$

□

Lemma 4.4.4. *$\preceq_f$ satisfies the NNRW property. Formally:*

Lemma 4.4.5. *$\preceq_f$ satisfies the NEU property. Formally:*

Lemma 4.4.6. *$\preceq_f$ refines any ranking function $\preceq$ satisfying LCE, NNRW, and NEU properties.*

Proof.

By definition of refinement, we need to show :

$p_x \preceq_f (p_y) \Rightarrow p_x \preceq (p_y)$

L.H.S. $= p_x \preceq_f (p_y)$

Using Theorem 1 on both sides. Theorem 1 is applicable because both $\preceq$ and $\preceq_f$ satisfy LCE and NNR

$\Leftrightarrow p_x \preceq_f (b^p|_{p_x}^{fl} + b^p|_{p_y}^{fl})\, p_y$

First, we show that we $\preceq_f$ is neutral, satisfy the limited collusion effect property and has a positive regard for wins. Then, we show that $\preceq_f$ is more refined than any other ranking function $\preceq_r$ that is neutral, satisfy the limited collusion effect property and has a positive regard for wins.

$\square$

4.5 Discussion

4.6 RelatedWork

Chapter 5

Conclusion and Future Work

REFERENCES

[1] Project Euler. Website. http://projecteuler.net/.

[2] The international SAT Competitions. Website. http://www.satcompetition.org/.

[3] Karim R Lakhani, Kevin J Boudreau, Po-Ru Loh, Lars Backstrom, Carliss Baldwin, Eric Lonstein, Mike Lydon, Alan MacCormack, Ramy A Arnaout, and Eva C Guinan. Prize-based contests can provide solutions to computational biology problems. *Nature Biotechnology*, 31(2):pp. 108–111, 2013.

[4] Jordi Petit, Omer Giménez, and Salvador Roura. Jutge.org: an educational programming judge. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, SIGCSE '12, pages 445–450, New York, NY, USA, 2012. ACM.

[5] TopCoder. The TopCoder Community. Website, 2009. http://www.topcoder.com/.