To appear in 8^{th} *International Conference on Computer Aided Verification*
Lecture Notes in Computer Science,
Springer-Verlag, July 1996.

Temporal Verification by Diagram Transformations*

Luca de Alfaro and Zohar Manna

Computer Science Department
Stanford University
Stanford, CA 94305, USA

Abstract. This paper presents a methodology for the verification of temporal properties of systems based on the gradual construction and algorithmic checking of *fairness diagrams.* Fairness diagrams correspond to abstractions of the system and its progress properties, and have a simple graphical representation.
In the proposed methodology, a proof of a temporal property consists of a chain of diagram transformations, starting from a diagram representing the original system and ending with a diagram that either corresponds directly to the specification, or that can be shown to satisfy it by purely algorithmic methods. Each diagram transformation captures a natural step of the gradual process of system analysis and proof discovery. The structure of fairness diagrams simplifies reasoning about progress properties, and the graphical representation provided by the diagrams enables the user to direct the construction of the proof. The resulting methodology is complete for proving specifications written in first-order linear-time temporal logic, provided no temporal operator appears in the scope of a quantifier.

1 Introduction

This paper presents a methodology for the verification of temporal properties of fair transition systems based on the gradual construction and algorithmic checking of *fairness diagrams*. Fairness diagrams represent abstractions of the system, and provide a graphical formalism for the study of its temporal properties. Fairness diagrams are graphs whose vertices are labeled by first-order assertions and whose edges are labeled by first-order transition relations. Their progress properties are represented by *fairness constraints*, which generalize the classical concepts of fairness [8].

In the proposed methodology, a proof of a temporal specification consists of a chain of diagram transformations, starting with a fairness diagram representing the original system and ending with a fairness diagram that either corresponds directly to the specification, or that can be shown to satisfy it by purely algorithmic methods. Since the transformations preserve containment of system

* This research was supported in part by the National Science Foundation under grant CCR-92-23226, the Advanced Research Projects Agency under NASA grant NAG2-892, the United States Air Force Office of Scientific Research under grant F49620-93-1-0139, and the Department of the Army under grant DAAH04-95-1-0317.

behaviors, the existence of this chain of transformations implies that the set of behaviors of the original system is a subset of the set of behaviors that satisfy the specification.

Fairness diagram transformations are intended to capture the step-by-step nature of the process of system analysis and proof construction. We introduce two types of transformations. The first type relies on the construction of simulation relations between diagrams, and provides a flexible way to analyze the safety properties of the system. The second type relies on the proof of new progress properties of a fairness diagram. Once proved, these properties can be represented as fairness constraints and added to the diagram. The form of the fairness constraints has been chosen to make it possible to use simple but complete rules to reason about progress properties. The resulting methodology is complete for proving specifications written in first-order linear-time temporal logic, provided no temporal operator appears in the scope of a quantifier.

Related work. Methods based on stepwise system transformations for the study of branching-time temporal properties of finite-state systems have been proposed in [3], and the use of simulation relations to study the temporal behavior of fair transition systems has been discussed in [4]. A related approach to the proof of temporal properties of systems is based on the use of verification diagrams [10, 1]. Like fairness diagrams, verification diagrams are graphs labeled with first-order assertions, and enable the proof of general temporal properties. Unlike fairness diagrams, verification diagrams represent a completed proof, and trade the advantage of gradual proof construction for conciseness of proof representation.

2 Fairness Diagrams

A *fairness diagram* (diagram, for short) $A = (\mathcal{V}, \Sigma, V, \rho, \tau, \Theta, \mathcal{F})$ consists of the following components:

1. A set $\mathcal{V}$ of typed variables, called *state variables.*
2. A *state space* Σ: each state $s \in \Sigma$ is a type-consistent value assignment of all the variables in $\mathcal{V}$. For $x \in \mathcal{V}$, we denote by $s(x)$ the value of x at state s.
3. A set V of *vertices.*
4. A mapping $\rho : V \mapsto 2^{\Sigma}$, that labels each vertex $v \in V$ with a subset $\rho(v) \subseteq \Sigma$. The set $\rho(v)$ represents the possible states of the system when the diagram is at vertex v, and is specified by a first-order formula $\widehat{\rho}(v)$ over $\mathcal{V}$, such that $\rho(v) = \{s \in \Sigma \mid s \models \widehat{\rho}(v)\}$.
5. A mapping $\tau : V^2 \mapsto 2^{\Sigma \times \Sigma}$, that labels each edge $(u, v) \in V^2$ with a relation $\tau(u, v) \subseteq \Sigma^2$. The relation is specified by a formula $\widehat{\tau}(u, v)$ over $\mathcal{V}, \mathcal{V}'$, such that $\tau(u, v) = \{(s, s') \mid (s, s') \models \widehat{\tau}(u, v)\}$, where (s, s') interprets $x \in \mathcal{V}$ as $s(x)$ and $x' \in \mathcal{V}'$ as $s'(x)$.
6. An initial region Θ. Regions are defined below.
7. A *fairness set* $\mathcal{F}$, defined below.

Locations and runs. A *location* of a diagram is a pair $(v, s) : v \in V, s \in \rho(v)$ composed of a vertex and of a corresponding state. We denote by $loc(A) = \{(v, s) \mid v \in V, s \in \rho(v)\}$ the set of all the locations of a diagram A. A location represents an instantaneous configuration of the diagram, similarly to a state of an FTS. A *run* of a diagram is an infinite sequence of locations (v_0, s_0), (v_1, s_1), (v_2, s_2), $\ldots$, such that $s_0 \in \Theta(v_0)$, and $(s_i, s_{i+1}) \in \tau(v_i, v_{i+1})$ for all $i \geq 0$.

Regions. A *region* Φ is a set of locations. We denote by $\Phi(v)$ the set of states $\{s \in \Sigma \mid (v, s) \in \Phi\}$ that are part of Φ at vertex v. A region Φ is represented by the set of formulas $\{\widehat{\Phi}(v)\}_{v \in V}$, where for each $v \in V$ the formula $\widehat{\Phi}(v)$ over $\mathcal{V}$ defines the set $\Phi(v)$. We say that Φ is an *integral region* if $\Phi(v)$ is equal to either $\emptyset$ or $\rho(v)$ for every $v \in V$. We can specify an integral region Φ by the set of vertices $\{v \in V \mid \Phi(v) \neq \emptyset\}$.

Modes. A *mode* $\lambda : V^2 \mapsto 2^{\Sigma \times \Sigma}$ labels each edge $(u, v) \in V^2$ with a transition relation $\lambda(u, v) \subseteq \tau(u, v)$. For $u, v \in V$, $\lambda(u, v)$ is represented by a formula $\widehat{\lambda}(u, v)$ over $\mathcal{V}$, $\mathcal{V}'$. A mode represents a subset of the possible transitions between locations of the diagram. An *integral mode* is a mode λ such that $\lambda(u, v)$ is either $\emptyset$ or $\tau(u, v)$, for all $u, v \in V$. We can specify an integral mode λ by listing the set of edges $\{(u, v) \mid \lambda(u, v) \neq \emptyset\}$.

Fairness constraints. A *fairness constraint* (constraint, for short) is a triple (J, C, G), where J, C are regions s.t. $C \subseteq J$ and G is a mode. Constraints are used to specify the fairness properties of the diagram, and the *fairness set* $\mathcal{F}$ is a set of constraints.

A diagram must satisfy the *consecution* condition, which states that if a transition is taken from a location, it will lead to another location: formally, for all $u, v \in V$ and for all $s, t \in \Sigma$,

$$s \in \rho(u) \wedge (s, t) \in \tau(u, v) \ \rightarrow \ t \in \rho(v) \ .$$

In the following, we denote by ϕ' the formula obtained from a first-order logic formula ϕ by replacing each free $x \in \mathcal{V}$ with $x' \in \mathcal{V}'$. With this convention, the consecution condition holds iff the logical implication $\widehat{\rho}(u) \wedge \widehat{\tau}(u, v) \rightarrow \widehat{\rho}'(v)$ is valid for all $u, v \in V$.

The computations of a diagram are defined in terms of its accepting runs.

Definition 1. A run $\sigma : (v_0, s_0)$, (v_1, s_1), $(v_2, s_2), \ldots$ of a diagram A is an *accepting run* if the following condition holds.

> *For each constraint $(J, C, G) \in \mathcal{F}$, if there is $n \geq 0$ such that $(v_i, s_i) \in J$ for all $i \geq n$ and $(v_i, s_i) \in C$ for infinitely many $i \geq 0$, then there are infinitely many $j \geq 0$ s.t. $(s_j, s_{j+1}) \in G(v_j, v_{j+1})$.*

If $\sigma : (v_0, s_0), (v_1, s_1), (v_2, s_2), \ldots$ is an accepting run of A, the sequence of states $s_0, s_1, s_2, \ldots$ is a *computation* of A. We denote by *Runs*(A), $\mathcal{L}(A)$ the sets of accepting runs and computations of A, respectively. ∎

According to the above definition, the informal reading of a constraint (J, C, G) is that every accepting run that stays in J forever and visits C infinitely often must follow a transition in G infinitely often. The chosen names J, C and G reflect the notions of *Justice* set, *Compassion* set and *Gratify* action that describe fairness of transition systems in [8].

A fair transition system (FTS), defined as in [9], can be represented by a diagram having only one vertex.

Construction 2 (from FTS to diagram). Let $S = (\mathcal{V}, \Sigma, \theta, \mathcal{T}, \mathcal{J}, \mathcal{C})$ be an FTS, where $\mathcal{V}$ is the set of state variables, Σ is the state space, $\theta \subseteq \Sigma$ is the initial condition, $\mathcal{T} = \{\gamma_1, \ldots, \gamma_m\}$ is the set of transitions, and $\mathcal{J} \subseteq \mathcal{T}$, $\mathcal{C} \subseteq \mathcal{T}$ are the just and compassionate transitions, respectively. The FTS can be represented by a diagram $fd(S) = (\mathcal{V}, \Sigma, V, \rho, \tau, \Theta, \mathcal{F})$, where $\mathcal{V}$, Σ are as in the FTS, $V = \{v_0\}$, $\rho(v_0) = \Sigma$, $\Theta(v_0) = \theta$, and τ and $\mathcal{F}$ are defined as follows.

1. $\tau(v_0, v_0) = \{(s, s) \mid s \in \Sigma\} \cup \bigcup_{i=1}^{m} \{(s, t) \in \Sigma^2 \mid t \in \gamma_i(s)\}$.
2. For $1 \leq i \leq m$, let $E_i(v_0) = \mathrm{Dom}(\gamma_i)$, $G_i(v_0) = \{(s, t) \in \Sigma^2 \mid t \in \gamma_i(s)\}$. If $\gamma_i \in \mathcal{J}$ (resp. $\gamma_i \in \mathcal{C}$) we add (E_i, E_i, G_i) (resp. $(loc(A), E_i, G_i)$) to $\mathcal{F}$. ■

We assume that an FTS S includes among its transitions the *idling* transition, that does not change the state. Given an FTS S, we will indicate with $\mathcal{L}(S)$ the computations admitted by S. Comparing the definitions of FTSs and diagrams, we have the following theorem.

Theorem 3. *For an FTS S, $\mathcal{L}(S) = \mathcal{L}(fd(S))$.*

3 Fairness Diagram Transformations

The temporal behavior of a diagram is studied by means of diagram transformations. These transformations preserve containment of behaviors, and they are reminiscent of the preorders of [3]. If a diagram A can be transformed into a diagram B by using one of the transformations, we write $A \Rightarrow B$. Since the transformations preserve containment of behaviors, $A \Rightarrow B$ implies $\mathcal{L}(A) \subseteq \mathcal{L}(B)$. We will denote by $\stackrel{*}{\Rightarrow}$ the reflexive transitive closure of $\Rightarrow$.

3.1 Simulation Transformations

Simulation transformations transform a diagram into a new one, such that the second diagram is capable of simulating the first one. These transformations modify the set of vertices of a diagram, rearranging the grouping of states in the vertices, and are used to study the safety properties of the diagram.

A *simulation relation* between two diagrams A_1 and A_2 is a function $V_1 \mapsto 2^{V_2}$ from the vertices of A_1 to those of A_2, which induces a simulation relation that maps each location (u, s) of A_1 into the subset $\bigcup_{v \in \mu(u)} (v, s)$ of locations of A_2. The following rule determines whether there is a simulation relation between two diagrams.

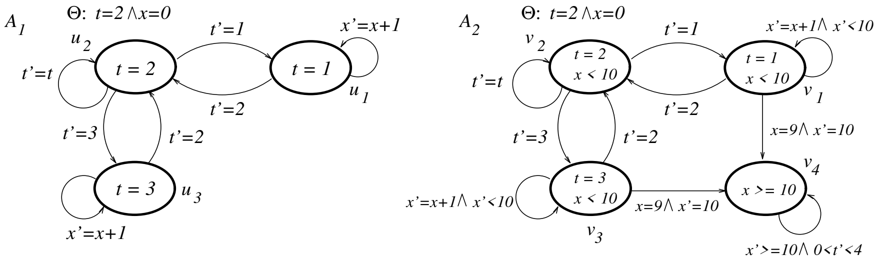

Fig. 1. Fairness diagram A_1 and A_2, related by the simulation relation arising from $\mu(u_1) = \{v_1, v_4\}$, $\mu(u_2) = \{v_2, v_4\}$, $\mu(u_3) = \{v_3, v_4\}$. Variables not mentioned in the transition relations are left unchanged.

Rule 4 (simulation). Let $A_1 = (\mathcal{V}, \Sigma, V_1, \rho_1, \tau_1, \Theta_1, \mathcal{F}_1)$, $A_2 = (\mathcal{V}, \Sigma, V_2, \rho_2, \tau_2, \Theta_2, \mathcal{F}_2)$ be two diagrams sharing the same variables and state space. We say that A_2 *simulates* A_1, written $A_1 \preceq A_2$, if there is a mapping $\mu : V_1 \mapsto 2^{V_2}$ such that the following logical assertions are valid.

1. For all $u \in V_1$, $\widehat{\Theta}_1(u) \to \bigvee_{v \in \mu(u)} \widehat{\Theta}_2(v)$.
2. For all $u, u' \in V_1$ and $v \in \mu(u)$, $\big(\widehat{\rho}_1(u) \wedge \widehat{\rho}_2(v) \wedge \widehat{\tau}_1(u, u')\big) \to \bigvee_{v' \in \mu(u')} \widehat{\tau}_2(v, v')$.
3. For each $(J_2, C_2, G_2) \in \mathcal{F}_2$ there is $(J_1, C_1, G_1) \in \mathcal{F}_1$ such that the following conditions are satisfied, for all $u \in V_1$ and $v \in \mu(u)$:
   (a) $\big(\widehat{J}_2(v) \wedge \widehat{\rho}_1(u)\big) \to \widehat{J}_1(u)$, and $\big(\widehat{C}_2(v) \wedge \widehat{\rho}_1(u)\big) \to \widehat{C}_1(u)$;
   (b) for all $u' \in V_1$, $\big(\widehat{\rho}_1(u) \wedge \widehat{\rho}_2(v) \wedge \widehat{G}_1(u, u')\big) \to \bigvee_{v' \in \mu(u')} \widehat{G}_2(v, v')$. ∎

Theorem 5. *If A_1, A_2 are two diagrams s.t. $A_1 \preceq A_2$, then $\mathcal{L}(A_1) \subseteq \mathcal{L}(A_2)$.*

Proof. Conditions 1 and 2 insure that for each run σ_1 of A_1 there is a related run σ_2 of A_2. Condition 3 insures additionally that if σ_1 is accepting, there is a related σ_2 of A_2 which is also accepting. The result then follows from the fact that the simulation relation is an identity with respect to the state space Σ. ∎

Transformation 6 (simulation transformation). Given two diagrams A_1, A_2, if $A_1 \preceq A_2$ we can transform A_1 into A_2. ∎

Example 7. Consider the diagrams A_1 and A_2 of Figure 1. With A_1 are associated the fairness constraints
$C_1^{(1)} = \big(\{u_1, u_2, u_3\}, \{u_1\}, \{(u_1, u_1)\}\big)$, $C_2^{(1)} = \big(\{u_1, u_2, u_3\}, \{u_3\}, \{(u_3, u_3)\}\big)$,
$C_3^{(1)} = \big(\{u_2\}, \{u_2\}, \{(u_2, u_1), (u_2, u_3)\}\big)$,
represented with the convention for integral regions and modes. With A_2 are associated the constraints

$$C_1^{(2)} = \big(\{v_1, v_2, v_3\}, \{v_1\}, \{(v_1, v_1), (v_1, v_4)\}\big),$$
$$C_2^{(2)} = \big(\{v_1, v_2, v_3\}, \{v_3\}, \{(v_3, v_3), (v_3, v_4)\}\big),$$
$$C_3^{(2)} = \big(\{v_2\}, \{v_2\}, \{(v_2, v_1), (v_2, v_3)\}\big).$$

Since the function μ of Figure 1 satisfies the conditions of Rule 4, A_1 can be transformed into A_2 using a simulation transformation. ∎

The proof of $A_1 \preceq A_2$ using Rule 4 fails if there is a constraint $(J_2, C_2, G_2) \in \mathcal{F}_2$ for which there is no constraint $(J_1, C_1, G_1) \in \mathcal{F}_1$ that satisfies conditions (3a) and (3b). In this case, to construct the simulation relation we must first add a suitable constraint to A_1. This can be done using fairness transformations.

3.2 Fairness Transformations

Fairness transformations analyze the structure of a diagram to derive new constraints that can be added to the set $\mathcal{F}$ without restricting the set of accepting runs of the diagram. These constraints are said to be *compatible*, and they represent progress properties of the diagram.

Definition 8. Given a diagram $A = (\mathcal{V}, \Sigma, V, \rho, \tau, \Theta, \mathcal{F})$ and a constraint (J, C, G), let $A' = (\mathcal{V}, \Sigma, V, \rho, \tau, \Theta, \mathcal{F} \cup \{(J, C, G)\})$ be the diagram obtained from A by adding (J, C, G) to the set $\mathcal{F}$. We say that the constraint (J, C, G) is *compatible* with A if $\mathit{Runs}(A) = \mathit{Runs}(A')$. ∎

Transformation 9 (fairness transformation). If diagram A' is obtained from diagram A by adding a compatible constraint, we can transform A into A'. ∎

To prove that a constraint is compatible with a diagram, we present verification rules. These rules are related to the rules for response and reactivity properties presented in [7]. The structure of the constraints enables two simplifications. First, separate rules for response and reactivity properties are not needed, since constraints can represent both types of properties. Second, it is possible to decompose the rules of [7] into simpler ones while retaining completeness.

We present three rules for proving the compatibility of constraints. The first rule shows the compatibility of a constraint independently from other constraints. The second rule uses one or two constraints in $\mathcal{F}$ to show that a third one is compatible, and can be thought as a rule to concatenate constraints. The third rule can be used to show that the union of constraints in $\mathcal{F}$ is compatible. Before presenting the rules, we introduce the notion of ranking functions.

Definition 10. A *well-founded domain* is a set D together with an order relation $>$, such that there is no infinite descending chain $d_0 > d_1 > d_2 > \ldots$ of elements of D. A *ranking function* $\delta : \mathit{loc}(A) \mapsto D$ for a diagram A is a function mapping pairs $(v, s) \in \mathit{loc}(A)$ into elements of a well-founded domain D. A ranking function δ is represented by the family of terms $\{\widehat{\delta}(u)\}_{u \in V}$ on $\mathcal{V}$, where term $\widehat{\delta}(u)$ denotes a function $\rho(u) \mapsto D$. ∎

Rule 11 (single constraint). A constraint (J, C, G) is compatible with the given diagram if there is a ranking function δ such that the assertions

$$\begin{aligned}
\widehat{J}(u) \wedge \widehat{\tau}(u, v) &\quad\rightarrow\quad \widehat{G}(u, v) \vee \widehat{\delta}(u) \geq \widehat{\delta}'(v) \vee \neg \widehat{J}'(v) \\
\widehat{C}(u) \wedge \widehat{\tau}(u, v) &\quad\rightarrow\quad \widehat{G}(u, v) \vee \widehat{\delta}(u) > \widehat{\delta}'(v) \vee \neg \widehat{J}'(v)
\end{aligned}$$

are valid for all $u, v \in V$. ∎

Justification. Assume that the conditions of the rule are satisfied and assume, towards the contradiction, that there is an accepting run σ that beyond position $k \geq 0$ stays forever in J and visits C infinitely often, without taking any transition in G. Beyond k, the value of δ along σ will not increase, and will decrease each time σ is in C. Since C is visited infinitely often, this contradicts the fact that the domain of δ is well-founded. ∎

Rule 12 (concatenation of constraints). A constraint (J, C, G) is compatible with the given diagram if there is a constraint $(J_0, C_0, G_0) \in \mathcal{F}$ with $\widehat{J}(u) \to \widehat{J}_0(u)$ for all $u \in V$ and a ranking function δ such that the following logical assertions are valid.

1. For all $u, v \in V$,

$$\begin{aligned} \widehat{J}(u) \wedge \widehat{\tau}(u,v) &\quad\to\quad \widehat{G}(u,v) \vee \neg\widehat{J}'(v) \vee \widehat{\delta}(u) \geq \widehat{\delta}'(v) \\ \widehat{J}(u) \wedge \widehat{G}_0(u,v) &\quad\to\quad \widehat{G}(u,v) \vee \neg\widehat{J}'(v) \vee \widehat{\delta}(u) > \widehat{\delta}'(v) \ . \end{aligned}$$

2. Either $\widehat{C}(u) \to \widehat{C}_0(u)$ for all $u \in V$, or there is $(J_1, C_1, G_1) \in \mathcal{F}$ such that for all $u, v \in V$, $\widehat{J}(u) \to \big[\widehat{J}_1(u) \vee \widehat{C}_0(u)\big]$, $\widehat{C}(u) \to \big[\widehat{C}_1(u) \vee \widehat{C}_0(u)\big]$, and

$$\widehat{J}(u) \wedge \widehat{G}_1(u,v) \quad\to\quad \widehat{G}(u,v) \vee \widehat{C}_0'(v) \vee \neg\widehat{J}'(v) \ . \quad ∎$$

Justification. Assume that the conditions of the rule are satisfied and assume, towards the contradiction, that beyond a certain position $k \geq 0$ an accepting run σ stays forever in J and visits C infinitely often without taking transitions in G. From the first condition of the rule, beyond position k the value of δ will not increase, and it will decrease whenever a transition in G_0 is taken. Thus, if we can prove that infinitely many transitions in G_0 are taken we reach the desired contradiction.

If $C \subseteq C_0$, this follows from $J \subseteq J_0$ and $(J_0, C_0, G_0) \in \mathcal{F}$. If $C \not\subseteq C_0$, the region $C - C_0$ is non-empty, and there are two cases. If σ visits infinitely often C_0, the result follows as before. If σ beyond position $j \geq 0$ visits infinitely often $C - C_0$ without entering C_0, then σ after j will be confined to J_1 and will visit C_1 infinitely often, and the result follows from $(J_1, C_1, G_1) \in \mathcal{F}$ and from the condition on G_1 in the rule. ∎

Rule 13 (union of constraints). Given n constraints $(J_1, C_1, G_1), \ldots, (J_n, C_n, G_n) \in \mathcal{F}$ and a region J_0, the constraint (J, C, G) defined by

$$J = J_0 \cup \bigcup_{i=1}^{n} J_i \qquad C = \bigcup_{i=1}^{n} C_i \qquad \forall u, v \in V : G(u,v) = \bigcup_{i=1}^{n} G_i(u,v)$$

is compatible with the given diagram if there is a ranking function δ such that the following assertions are valid, for $1 \leq i \leq n$:

$$\begin{aligned} \widehat{J}_i(u) \wedge \widehat{\tau}(u,v) \wedge \widehat{J}_i'(v) &\quad\to\quad \widehat{G}(u,v) \vee \widehat{\delta}(u) \geq \widehat{\delta}'(v) \\ \widehat{J}_i(u) \wedge \widehat{\tau}(u,v) \wedge \neg\widehat{J}_i'(v) &\quad\to\quad \widehat{G}(u,v) \vee \widehat{\delta}(u) > \widehat{\delta}'(v) \vee \neg\widehat{J}'(v) \\ \widehat{J}_0(u) \wedge \widehat{\tau}(u,v) &\quad\to\quad \widehat{G}(u,v) \vee \widehat{\delta}(u) \geq \widehat{\delta}'(v) \vee \neg\widehat{J}'(v) \ . \end{aligned} \quad ∎$$

Justification. Assume that the conditions of the rule are satisfied and, towards the contradiction, that beyond a certain position $k \geq 0$ an accepting run σ stays forever in J visiting C infinitely often without taking transitions in G. As beyond k the value of δ never increases, and decreases every time σ leaves a region J_i, $1 \leq i \leq n$, σ can leave only finitely many times every J_i. Since $C_i \subseteq J_i$ for all $1 \leq i \leq n$ and σ visits infinitely often $\bigcup_{i=1}^{n} C_i$, there must be $m \in [1..n]$ s.t. eventually σ is confined to J_m and visits C_m infinitely often. The contradiction then follows from the fact that $(J_m, C_m, G_m) \in \mathcal{F}$. ∎

From the justifications of the rules, we have the following theorem.

Theorem 14 (soundness). *If the conditions of each of the rules 11, 12 or 13 are satisfied, the constraint (J, C, G) is compatible with the diagram under consideration.*

Example 15. Consider diagram A_2 of Example 7. Using Rule 13, it is possible to add to it the compatible constraint

$$C_4^{(2)} = (\{v_1, v_2, v_3\}, \{v_1, v_3\}, \{(v_1, v_1), (v_1, v_4), (v_3, v_3), (v_3, v_4)\})$$

resulting from the union of $C_1^{(2)}$ and $C_2^{(2)}$. By Rule 12, with $C_4^{(2)}$ for (J_0, C_0, G_0) $C_3^{(2)}$ for (J_1, C_1, G_1), and ranking function $\widehat{\delta}(v_1) = \widehat{\delta}(v_2) = \widehat{\delta}(v_3) = 10 - x$, $\widehat{\delta}(v_4) = 0$, it is possible to add the constraint

$$C_5^{(2)} = (\{v_1, v_2, v_3\}, \{v_1, v_2, v_3\}, \{(v_1, v_4), (v_3, v_4)\}).$$

Let A_2' be the diagram obtained by adding $C_4^{(2)}$ and $C_5^{(2)}$ to A_2. Intuitively, $C_5^{(2)}$ represent the temporal progress property $\diamond(x \geq 10)$, satisfied by A_2'. ∎

3.3 Completeness and Complexity Results

The transformations introduced in the previous sections are complete for proving the compatibility of fairness constraints, as the following theorem states.

Theorem 16 (completeness for constraints). *Given a diagram A and a constraint (J, C, G), if (J, C, G) is compatible there is a sequence of transformations $A \overset{*}{\Rightarrow} B$, where the diagram B is obtained from A by adding (J, C, G) to $\mathcal{F}$.*

The proof of this theorem is rather lengthy, and follows the general line of the completeness proof for reactivity and response rules presented in [7]. The complete proof is given in [2]. To state the completeness theorem for transition systems we need an additional definition.

Definition 17. A diagram $A = (\mathcal{V}, \Sigma, V, \rho, \tau, \Theta, \mathcal{F})$ is *state-deterministic* if for all $u, v, w \in V$ with $v \neq w$ it is $\Theta(v) \cap \Theta(w) = \emptyset$ and $\tau(u, v) \cap \tau(u, w) = \emptyset$. ∎

Theorem 18 (completeness for transition systems). *Let $A = fd(S)$ for an FTS S, and B be a state-deterministic diagram. If $\mathcal{L}(S) \subseteq \mathcal{L}(B)$, there is a chain of transformations $A \overset{*}{\Rightarrow} B$.*

The proof of this theorem relies on Theorem 16 for the fairness part, and follows otherwise from the existence of chains of simulation relations between diagrams derived from FTSs and deterministic diagrams.

Complexity of transformations. To establish a simulation transformation $A_1 \Rightarrow A_2$ using Rule 4, the number of logical formulas to be considered is $O(|V_1|^2 \cdot |V_2|)$, where V_1, V_2 are the set of vertices of A_1, A_2 respectively.

To add a constraint to a diagram A, the number of logical formulas to be considered is $O(|V|^2)$ using Rules 11, 12, and $O(n|V|^2)$ using Rule 13, where n is the number of constraints whose union is taken. These bounds, however, refer to the worst-case complexity. If these transformations are used to do a local analysis of a diagram that involves only few vertices, the number of non-trivial logical formulas to be proved does not necessarily increase when the size of the diagram increases.

4 Proving Linear Temporal Logic Properties

Let TL_s be the class of temporal formulas obtained by combining first-order logic formulas using propositional connectives, the future temporal operators $\bigcirc$ (next), $\Box$ (always), $\Diamond$ (eventually), $\mathcal{U}$ (until), and of the corresponding past ones $\ominus$, $\boxminus$, $\diamondminus$ and $\mathcal{S}$ [8]. Note that in a formula $\phi \in TL_s$, no temporal operator occurs in the scope of a quantifier.

Given an FTS S and $\phi \in TL_s$, in this section we present two methods for proving that all computations of S satisfy ϕ, written $S \models \phi$. According to the first method, we construct from ϕ a deterministic Streett automaton M_ϕ, we translate it into a diagram $fd(M_\phi)$, and we show that $fd(S) \stackrel{*}{\Rightarrow} fd(M_\phi)$. According to the second method, we construct a nondeterministic Streett automaton $N_{\neg\phi}$ representing $\neg\phi$ and we show that $fd(S) \stackrel{*}{\Rightarrow} B$, where B is a diagram s.t. $\mathcal{L}(B) \cap \mathcal{L}(N_{\neg\phi}) = \emptyset$ can be shown using algorithmic methods. The Streett automata used in the above methods are a first-order version of the classical ones [11].

Definition 19 (Streett automaton). A (first-order) *Streett automaton* A consists of the components $(\mathcal{V}, \Sigma, (V, E), \rho, Q, \mathcal{A})$, where $\mathcal{V}$, Σ, ρ are as in a diagram; (V, E) is a directed graph with set of vertices V and set of edges $E \subseteq V^2$; $Q \subseteq V$ is the set of *initial vertices*, and $\mathcal{A}$, called the *acceptance list*, is a set of pairs $(P, R) : P, R \subseteq V$.

A run σ of A is an infinite sequence of locations $(v_0, s_0), (v_1, s_1), (v_2, s_2), \ldots$ such that $v_0 \in Q$ and $s_i \in \rho(v_i)$, $(v_i, v_{i+1}) \in E$ for all $i \geq 0$. Run σ is an accepting run of A if the following condition holds:

> *For each pair $(P, R) \in \mathcal{A}$, either $v_i \in R$ for infinitely many $i \in \mathbb{N}$, or there is $k \in \mathbb{N}$ such that $v_i \in P$ for all $i \geq k$.*

The set of accepting runs (resp. computations) of a Streett automaton A is denoted by $Runs(A)$ (resp. $\mathcal{L}(A)$). ∎

Given a Streett automaton M, we can construct a fairness diagram $fd(M)$ such that $\mathcal{L}(fd(M)) = \mathcal{L}(M)$.

Construction 20 (from Streett Automaton to diagram). Given a Streett automaton $M = (\mathcal{V}, \Sigma, (V, E), \rho, Q, \mathcal{A})$ we can construct a fairness diagram $fd(M) = (\mathcal{V}, \Sigma, V, \rho, \tau, \Theta, \mathcal{F})$ as follows.

1. For all $u, v \in V$, $\tau(u,v) = \rho(u) \times \rho(v)$ if $(u,v) \in E$, and $\tau(u,v) = \emptyset$ otherwise.
2. $\Theta = \{(u,s) \mid u \in Q \wedge s \in \rho(u)\}$.
3. $\mathcal{F}$ consists of all constraints (J, C, G) such that there is $(P, R) \in \mathcal{A}$ for which $J = C = \{(u,s) \mid u \in V - P \wedge s \in \rho(u)\}$, and for all $u, v \in V$, $G(u,v) = \tau(u,v)$ if $v \in R$, and $G(u,v) = \emptyset$ otherwise. ∎

4.1 The Transformation Method

For a temporal logic formula $\phi \in TL_s$, let $\mathcal{L}(\phi)$ be the set of computations that satisfy ϕ. Let M_ϕ be a deterministic Streett automaton such that $\mathcal{L}(M_\phi) = \mathcal{L}(\phi)$. This automaton can be constructed from ϕ with the methods explained in [6, 11, 12]. We can thus formulate the first proof strategy.

Proof Strategy 1. To prove $S \models \phi$ for FTS S and a formula $\phi \in TL_s$, construct a chain of transformations $fd(S) \stackrel{*}{\Rightarrow} fd(M_\phi)$. ∎

Theorem 21. *Proof Strategy 1 is sound and complete for proving $S \models \phi$ for an FTS S and $\phi \in TL_s$.*

Proof. The soundness result follows from $\mathcal{L}(S) = \mathcal{L}(fd(S)) \subseteq \mathcal{L}(fd(M_\phi)) = \mathcal{L}(\phi)$. Since M_ϕ is deterministic, $fd(M_\phi)$ is state-deterministic, and the completeness result follows from Theorem 18. ∎

The drawback of Strategy 1 is that, in the worst case, the number of vertices of M_ϕ is doubly exponential in the size $|\phi|$ of the specification ϕ.

4.2 The Product Method

Given a temporal formula $\phi \in TL_s$, it is possible to construct a nondeterministic Streett automaton $N_{\neg\phi}$ s.t. $\mathcal{L}(N_{\neg\phi}) = \mathcal{L}(\neg\phi)$. The automaton $N_{\neg\phi}$ has number of vertices that is singly exponential in $|\phi|$. To prove $\mathcal{L}(S) \subseteq \mathcal{L}(\phi)$ for an FTS S, it suffices to construct a chain of transformations $fd(S) \stackrel{*}{\Rightarrow} B$ ending with a diagram B s.t. $\mathcal{L}(B) \cap \mathcal{L}(fd(N_{\neg\phi})) = \emptyset$ can be shown with algorithmic methods. The emptiness problem of $\mathcal{L}(A) \cap \mathcal{L}(B)$ for diagrams A, B is undecidable: in the following, we give a computable sufficient condition for $\mathcal{L}(A) \cap \mathcal{L}(B) = \emptyset$.

Let FL be the first-order logic language with interpreted function and predicate symbols in which the assertions labeling the diagrams and the first-order part of the temporal specifications are written. Assume that we have a proof procedure $\vdash$ for FL that always terminates, and that is able to prove a subset of the valid sentences that includes all substitution instances of propositional tautologies. Given $\phi \in FL$, if $\vdash$ terminates with a proof of ϕ we write $\vdash \phi$; otherwise we write $\not\vdash \phi$. To obtain a sufficient condition for the emptiness of the intersection of diagram languages, we construct the *graph product* of the diagrams.

Construction 22 (graph product of diagrams). Given two diagrams $A_1 = (\mathcal{V}, \Sigma, V_1, \rho_1, \tau_1, \Theta_1, \mathcal{F}_1)$, $A_2 = (\mathcal{V}, \Sigma, V_2, \rho_2, \tau_2, \Theta_2, \mathcal{F}_2)$ their *graph product* $A_1 \otimes A_2 = ((V, E), V_{in}, \mathcal{G})$ consists of a graph (V, E), of a subset $V_{in} \subseteq V$ of initial vertices, and of a set $\mathcal{G}$ of triples of the form $(P, Q, R) : P, Q \subseteq V, R \subseteq V^2$. These components are defined in terms of the components of A_1 and A_2 as follows.

1. $V = \{(v_1, v_2) \in V_1 \times V_2 \mid \not\vdash \neg(\widehat{\rho}(v_1) \wedge \widehat{\rho}(v_2))\}$.
2. $V_{in} = \{(v_1, v_2) \in V \mid \not\vdash \neg(\widehat{\Theta}_1(v_1) \wedge \widehat{\Theta}_2(v_2))\}$.
3. $E = \{((u_1, u_2), (v_1, v_2)) \in V^2 \mid \not\vdash \neg(\widehat{\tau}_1(u_1, v_1) \wedge \widehat{\tau}_2(u_2, v_2))\}$.
4. For $i = 1, 2$, to each constraint $(J, C, G) \in \mathcal{F}_i$ corresponds the triple $(\kappa(J, i), \kappa(C, i), \pi(G, i))$, where κ and π are defined by:
$$\kappa(\Phi, i) = \{(u_1, u_2) \in V \mid \vdash (\widehat{\Phi}(u_i) \leftrightarrow \widehat{\rho}_i(u_i))\}$$
$$\pi(\lambda, i) = \{((u_1, u_2), (v_1, v_2)) \in E \mid \not\vdash \neg\widehat{\lambda}(u_i, v_i)\} ,$$
for all regions Φ and modes λ of A_i. The set $\mathcal{G}$ is then
$$\mathcal{G} = \{(\kappa(J, i), \kappa(C, i), \pi(G, i)) \mid 1 \le i \le 2 \wedge (J, C, G) \in \mathcal{F}_i\} . \quad \blacksquare$$

Theorem 23. *Given two diagrams A, B, let $A \otimes B = ((V, E), V_{in}, \mathcal{G})$. A sufficient condition for $\mathcal{L}(A) \cap \mathcal{L}(B) = \emptyset$ is that for each strongly connected component $U \subseteq V$ of (V, E) reachable from V_{in} there is $(P, Q, R) \in \mathcal{G}$ s.t. $U \subseteq P$, $U \cap Q \neq \emptyset$, and $(u, v) \notin R$ for all $u, v \in U$.*

Proof Strategy 2. To prove $S \models \phi$ for an FTS S and a formula $\phi \in TL_s$, construct a chain of transformations $fd(S) \stackrel{*}{\Rightarrow} B$ to a diagram B s.t. $\mathcal{L}(B) \cap \mathcal{L}(fd(N_{\neg\phi})) = \emptyset$ can be proved using the condition of Theorem 23. $\blacksquare$

Example 24. If $\phi : \Diamond(x \ge 10)$, the Streett automaton $N_{\neg\phi}$ will consist of only one vertex, labeled with $x < 10$. Using Theorem 23, it is easy to check that the graph product of $N_{\neg\phi}$ and diagram A'_2 of Example 15 has empty language. $\blacksquare$

Theorem 25. *Proof Strategy 2 is sound and complete for proving $S \models \phi$ for an FTS S and $\phi \in TL_s$.*

Proof. The soundness part is a consequence of the previous definitions. Let $fd(M_\phi)$ be the deterministic diagram corresponding to ϕ, as in the previous strategy. If $S \models \phi$, there is a chain of transformations $fd(S) \stackrel{*}{\Rightarrow} fd(M_\phi)$, and from the construction of $fd(M_\phi)$ it can be seen that the graph product $fd(M_\phi) \otimes fd(N_{\neg\phi})$ satisfies the condition for emptiness of Theorem 23. $\blacksquare$

Note that we still need a complete deductive system for the interpreted first-order language FL to perform the diagram transformations, in order to retain the completeness results expressed by Theorems 16, 18 and 25.

From the above proof, we see that there is a final diagram B for strategy 2 with number of vertices at most doubly exponential in $|\phi|$. In fact, if the state space Σ of the FTS if finite, it is possible to show that there is a diagram B with number of vertices bound by $|\Sigma|$, so that the number of vertices of $B \otimes fd(N_{\neg\phi})$ is at most singly exponential in $|\phi|$, similarly to the case of finite-state model checking.

For systems with an infinite number of reachable states, the worst-case complexity of strategy 2 is not better than the one of Proof Strategy 1. However, in most cases an FTS S will satisfy a specification ϕ by exhibiting a set of computations $\mathcal{L}(S)$ significantly smaller than $\mathcal{L}(\phi)$. Thus, the diagram B of Proof Strategy 2 in general is smaller than $fd(M_\phi)$, so that Proof Strategy 2 is often more convenient than Proof Strategy 1.

5 Conclusions

Fairness diagrams provide a methodology for the proof of temporal specifications of systems. They can also be used as a graphical specification language. Since both vertices and edges are labeled with first-order assertions, fairness diagrams have the advantage over traditional temporal logic (and similarly to TLA [5]) of providing a simpler representation for specifications that involve conditions on both system states and actions.

While we have given completeness results on the existence of chains of transformations, we have not discussed how to obtain guidance for their construction. When the specification has a simple temporal form, the graphical representation of the diagrams often captures enough intuition about the system to guide the construction of the chain of transformations. We intend to address the question of guidance and heuristics for chain constructions in future work.

Acknowledgements. We would like to thank Anca Browne, Henny Sipma and Tomás Uribe for helpful comments and suggestions.

References

1. I.A. Browne, Z. Manna, and H.B. Sipma. Generalized verification diagrams. In *Found. of Software Technology and Theoretical Comp. Sci.*, volume 1026 of *Lect. Notes in Comp. Sci.*, pages 484–498. Springer-Verlag, 1995.
2. L. de Alfaro and Z. Manna. Temporal verification by diagram transformations. Technical report, Stanford University, 1996.
3. O. Grumberg and D.E. Long. Model checking and modular verification. *ACM Trans. Prog. Lang. Sys.*, 16(3):843–871, May 1994.
4. Y. Kesten, Z. Manna, and A. Pnueli. Temporal verification of simulation and refinement. In *Proc. of the REX Workshop "A Decade of Concurrency"*, volume 803 of *Lect. Notes in Comp. Sci.*, pages 273–346. Springer-Verlag, 1994.
5. L. Lamport. The temporal logic of actions. *ACM Trans. Prog. Lang. Sys.*, 16(3):872–923, May 1994.
6. O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specifications. In *Proc. 12th ACM Symp. Princ. of Prog. Lang.*, pages 97–107, 1985.
7. Z. Manna and A. Pnueli. Completing the temporal picture. *Theor. Comp. Sci.*, 83(1):97–130, 1991.
8. Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification.* Springer-Verlag, New York, 1991.
9. Z. Manna and A. Pnueli. Models for reactivity. *Acta Informatica*, 30:609–678, 1993.
10. Z. Manna and A. Pnueli. Temporal verification diagrams. In *TACS 94*, Lect. Notes in Comp. Sci. Springer-Verlag, 1994.
11. S. Safra. On the complexity of ω-automata. In *Proc. 29th IEEE Symp. Found. of Comp. Sci.*, 1988.
12. S. Safra and M.Y. Vardi. On ω-automata and temporal logic. In *Proc. 21th ACM Symp. Theory of Comp.*, pages 127–137, 1989.