ISSN 1674-3202

逻辑学研究

Studies in Logic

第 2 卷，第 4 期，冬季号，2009 年 12 月
Volume 2, Number 4 , Winter , December 2009

ISSN 1674-3202

逻辑学研究

Studies in Logic

第 2 卷，第 4 期，冬季号，2009 年12月
Volume 2, Number 4, Winter, December

中山大学　中国逻辑学会　主办
Sponsored by Sun Yat-sen University
and The Chinese Association of Logic

逻辑学研究（季刊）

逻辑学研究（季刊）
第 2 卷, 第 4 期, 总第 7 期
2009 年第 4 期，12 月 18 日出版

目　　录

一个用于表达因果关系的ATL的扩展　刘虎　1
因果推理中的亚决定性　张寄冀　16
汉语语用标记语与规约含义　冯光武　48
模态逻辑典范框架的生成子框架　裘江杰　75
论模态逻辑的集合论语义　史璟　82
塔斯基：语义性真理论与符合论　李主斌　97

第八届逻辑与认知国际会议报道　沈榆平 114
“纪念中国逻辑学会成立30周年大会”记略　杜国平，邹崇理　120

Studies in Logic
Volume 2, Number 4, December 2009

TABLE OF CONTENTS

An Extension of ATL for Model Checking Causality 1
Hu Liu

Underdetermination in Casual Inference 16
Jiji Zhang

Chinese Pragmatic Markers and Conventional Implicature 48
Guangwu Feng

The Generated Sub-frame of Canonical Frame of Modal Logic 75
Jiangjie Qiu

On Set-theoretic Semantics for Modal Logic 82
Jing Shi

Tarski: Semantic Truth and the Correspondence Theory 97
Zhubin Li

Conference Report: on the 8th International Conference on Logic and Cognition 114
Yuping Shen

Conference Report: the Conference of 30^{th} Anniversary of the Chinese Association of Logic in Chengdu 120
Guoping Du, Chongli Zou

Studies in Logic, Vol. 2, No. 4 (2009): 1–15
PII: 1674-3202(2009)-04-0001-15

An Extension of ATL for Model Checking Causality*

Hu Liu
Institute of Logic and Cognition, Sun Yat-sen University
liuhu2@mail.sysu.edu.cn

Abstract. CTL model checking has been widely used in formal specification and verification. The so called alternating time temporal logic (ATL) is an extension of CTL to formalize a game-like multiplayer specification. The property that players can guarantee desired system state is modeled in ATL by using cooperation operators. In practice, it is usually important to know that a causal link indeed exists between players' actions and states of a system, especially when modularity is considered. We extend ATL by introducing new operators to characterize causalities. The resultant logics are more expressive than ATL, while share the same computational complexity.

1. Introduction

Techniques for automatic formal verification of finite state transition systems have developed in the last two decades. The most successful of these methods is called model checking [3]. Computational systems are formalized by formulas of branching time temporal logics, among which the *computation tree logic* (CTL) is the most widely used formalism. A finite state transition system is transformed to a branching time structure by interleaving the states transition relation of the system. Each path of the resultant branching time structure is a possible computation of the system.

CTL uses temporal modalities $\bigcirc$ ("next"), $\Box$ ("always"), and $\mathcal{U}$ ("until") together with a path quantifier E to express that a certain property of a system is satisfied in some possible computation. For example, $E\Box\varphi$ means that there exists a computation such that φ is always true along this computation. The problem of CTL model checking is to determine whether a CTL formula is satisfiable in a given state of a states transition system. There exist efficient CTL model checkers, such as SMV [7], that have been used in industrial practice for normal quality control process [4].

An extension of CTL, called *alternating-time temporal logic* (ATL), has been proposed to tackle the specification and verification of open systems [1]. An important model checking problem of open systems is so-called *alternating satisfaction*: A group of processes can guarantee a system entering into certain states, no matter what other processes do. ATL is defined on a game-like structure. Each process is

Received 2009-10-20
*This paper was supported by A Foundation for the Author of National Excellent Doctoral Dissertation of PR China (FANEDD 2007B01).

called a player. A set of cooperation operators $\langle\langle A\rangle\rangle$, where A is a set of players, is present to characterize game-like properties. A formula of the form $\langle\langle A\rangle\rangle\varphi$ is read as that "players in A can guarantee that φ is true" or equivalently, "players in A have a collective winning strategy to bring about φ regardless of what other players of the system do." ATL is an extension of CTL because the path quantifier E of CTL is definable in ATL by $\langle\langle\Sigma\rangle\rangle$, where Σ is the set of all players. The dual quantifier A of E is defined by $\langle\langle\emptyset\rangle\rangle$. Model checking problem of ATL is proved to be PTIME-complete [1].

In ATL, each player is endowed with a set of strategies. Suppose that at a state q the set of all possible computations is C. when a set of players A have chosen their strategies F_A, one for each player in A, C is limited by F_A to C' with $C' \subseteq C$. In other words, players A act in a certain way and rule out computations in C/C'. Formula $\langle\langle A\rangle\rangle\psi$ is true at state q if and only if such collective strategies F_A exists that ψ is true at all computations in C' limited by F_A.

Although at first sight, alternating satisfaction implies that it is players' activities that cause certain system state, it in fact makes no such promise. $\langle\langle A\rangle\rangle\psi$ can be true without any causal link between A' actions and truth of ψ. An obvious counterexample is when ψ is a valid formula. In this case $\langle\langle A\rangle\rangle\psi$ is true because ψ is true everywhere. A has nothing to do with the truth value of ψ.

We shall call it *alternating causality* that a group of players *cause* a system entering into certain states, no matter what other processors do. It is different from alternating satisfaction in just one word, "guarantee" being replaced by "cause" to indicate existence of a causal link. Alternating causality is an important property of open systems, especially when modularity is considered. Suppose that a bundle of processes A is designed as an unit of anti-deadlock. Even if $\langle\langle A\rangle\rangle(\Box\ No\_deadlock)$ is checked to be true at desired states, it is not safe to say that A functions well, because there is a lack of causality to indicate that $No\_deadlock$ is a result of A's moves. It may come from other parts of the system. In that case A may fail to function of $No\_deadlock$ in a different environment.

We will consider two different but related types of connections between players' activities and system's states. First, we consider players' power of control over states, or syntactically, players' power of control over truth values of formulas. A memory allotment unit may have full power over states of system memory, and less power over states of system processor, and even less power or no power over states of network access. For example, A may have full power over the values of a variable x, and less power over the values of $x \vee y$, since y's values may depend on other players.

The second issue concerns relations among players. We discuss the exclusivity property and minimality of player in performing a function. Intuitively, both power of control and exclusivity matter with causalities, and they are closely related. We will show that a full power of control implies exclusivity.

The remainder of the paper is structured as follows. Next section is an intro-

duction to game structure and ATL. Section 3 and 4 are the principal parts consisting of both informal discussions and formal definitions of CATL and SATL. We present symbolic model checking procedures in section 5.

2. Alternating Time Temporal Logic

Open systems are modelled by (concurrent) game structures, which reflect simultaneous steps by the players. Definitions in this section are mainly from [1] and [5].

Definition 2.1 *A* game structure *is a tuple* $S = (\Sigma, Q, \Pi, \pi, \delta)$ *, where*

(1) $\Sigma = \{a_1, \ldots, a_n\}$ *is a finite set of* players;
(2) Q *is a finite, non-empty set of* states;
(3) Π *is a finite set of* propositions;
(4) For each state $q \in Q$, $\pi(q) \subseteq \Pi$ *is a set of propositions true at* q. *The function* π *is called a labelling function;*
(5) $\delta : \Sigma \times Q \rightarrow 2^{2^Q}$ *is the system transition function, which maps each pair of player and state to a set of sets of states, with* $\delta(a, q)$ *reflecting the set of choices available to the player* a *when the system is in the state* q. *We require that this function satisfy the constraint that the system is completely controlled by its component players: For every state* q, $\bigcap\{Q_{a_1}, \ldots, Q_{a_n}\}$ *is a singleton, where for each player* $a_i, Q_{a_i} \in \delta(a_i, q)$ *is a possible choice for* a_i *at* q. *That means that if every player has made her choice, the system is completely determined.*

The *alternating-time temporal logic* (ATL for short) is defined with respect to a finite set Π of propositions and a finite set Σ of players. An ATL formula is one of the following:

(S1) p, for propositions $p \in \Pi$.
(S2) $\neg\varphi$ or $\varphi_1 \vee \varphi_2$, where φ, φ_1 and φ_2 are ATL formulas.
(S3) $\langle\langle A\rangle\rangle \bigcirc \varphi$, $\langle\langle A\rangle\rangle \Box\varphi$, or $\langle\langle A\rangle\rangle \varphi_1 \mathcal{U} \varphi_2$, where $A \subseteq \Sigma$ is a set of players, and φ, φ_1, and φ_2 are ATL formula.

Additional Boolean connectives are defined from $\neg$ and $\vee$ in the usual manner. Operator $\langle\langle A\rangle\rangle$ is a path quantifier, $\bigcirc$ ("next"), $\Box$ ("always"), and $\mathcal{U}$ ("until") are temporal operators. We write $\langle\langle A\rangle\rangle \Diamond\varphi$ for $\langle\langle A\rangle\rangle true\, \mathcal{U}\varphi$.

For two states q, q' and a player a, we say that q' is an a-*successor* of q if there is a set $Q' \in \delta(a, q)$ such that $q' \in Q'$. Intuitively, if q' is an a-successor of q, then q' is a possible outcome of one of the choices available to a at q. We denote by $succ(a, q)$ the set of a-successors at state q. We say that q' is a *successor* of q if for each player $a \in \Sigma$, we have $q' \in succ(a, q)$; intuitively, if q' is a *successor* of q,

then when the system is at state q, the players Σ can cooperate to ensure that q' is the next state.

A *computation* is an infinite sequence of states $\lambda = q_0, q_1, \ldots$ such that for all $i \geq 0$, the state q_{i+1} is a successor of q_i. A computation starting at state q is referred to as a *q-computation*; $\lambda[i]$ denotes q_i in the sequence λ; $\lambda[0, i]$ and $\lambda[i, \infty]$ denotes the finite prefix $q_0, \ldots, q_i$ and the infinite suffix $q_i, q_{i+1}, \ldots$ respectively.

Let Q^+ be the set of all non-empty finite sequences over Q. A strategy f_a for a player a is a function $f_a : Q^+ \rightarrow 2^Q$ such that if the last state of a sequence λ is q, then $f_a(\lambda) \in \delta(a, q)$. Thus, a strategy f_a determines for each finite sequence an a's choice as her next action. Each strategy f_a induces a set of computations that player a can enforce. Suppose $A \subseteq \Sigma$ is a set of agents and $F_A = \{f_a \mid a \in A\}$ is a set of strategies, one for each player in A. For a state q we define the outcomes of F_A from q to be the set $out(q, F_A)$ of the q-computations that the players in A enforce when they follow the strategies in F_A; that is, a computation $\lambda = q_0, q_1, \ldots$ is in $out(q, F_A)$ if $q_0 = q$ and for all positions $i \geq 0$, $q_{i+1} \in \bigcap\{f_a(\lambda[0, i]) \mid f_a \in F_A\}$.

For convenience we define satisfaction of a temporal formula at a computation λ as follows. Note that a temporal formula is not an ATL formula. We use ψ to range over temporal formulas.

$\lambda \models \bigcirc\varphi$ iff $\lambda[1] \models \varphi$.
$\lambda \models \Box\varphi$ iff $\lambda[i] \models \varphi$ for all $i \geq 0$.
$\lambda \models \varphi_1 \mathcal{U} \varphi_2$ iff there exists a $i \geq 0$ such that $\lambda[i] \models \varphi_2$ and for all $0 \leq j < i$, we have $\lambda[j] \models \varphi_1$.

We write $S, q \models \varphi$ to indicate that φ is satisfied at state q in game structure S. When S is clear from the context, we omit it and write $q \models \varphi$. The semantic rules for the satisfaction relation $\models$ are defined as follows:

$q \models p$ iff $p \in \pi(q)$, where p is a proposition.
$q \models \neg\varphi$ iff $q \not\models \varphi$.
$q \models \varphi_1 \vee \varphi_2$ iff $q \models \varphi_1$ or $q \models \varphi_2$.
$q \models \langle\langle A \rangle\rangle \psi$, where $\psi \in \{\bigcirc\varphi, \Box\varphi, \varphi_1 \mathcal{U} \varphi_2\}$ iff there exists a set of strategies F_A, one for each $a \in A$, such that for all computations $\lambda \in out(q, F_A)$, we have $\lambda \models \psi$.

ATL is an extension of CTL. [1] shows that the CTL path quantifier E can be look upon as a special cooperation operator $\langle\langle \Sigma \rangle\rangle$ in ATL, where Σ is the set of all players.

3. CATL

$\langle\langle A \rangle\rangle \psi$ is understood by saying that ψ is guaranteed to be true by A. It states that ψ is true along all paths selected by a collective strategy of A. The strategy

does not necessarily have real effect on ψ. ψ being a valid formula is an obvious counterexample. In this case A has absolutely no power of control over ψ.

Researchers in *stit* theory has already noticed the problem. Based on different structures, *stit* operator has some similarities with cooperation operators. Readers may consult [2] for an overview of the area. Horty introduced in [6] a *dstit* operator to eliminate the problem. Extending the idea to ATL, we have an alternative operator $\langle\langle A\rangle\rangle'$ as defined as follows.

> $q \models \langle\langle A\rangle\rangle'\psi$, where ψ is a temporal formula, iff there exists a set of strategies F_A, one for each $a \in A$, such that for all computations $\lambda \in out(q, F_A)$, we have $\lambda \models \psi$, and (2) there exists a q-computation λ such that $\lambda \not\models \psi$.

The positive condition (1) is the same in the definition of $\langle\langle A\rangle\rangle$. The negative condition (2) states that ψ is not a settled truth yet, so that A's action may have some real effect on ψ. $\langle\langle A\rangle\rangle'\psi$ indicates than A has more power of control over ψ than $\langle\langle A\rangle\rangle\psi$ does.

dstit operator is not definable by *stit*. To do so would need an overall path quantifier. Such quantifier is definable in ATL. The following validity is a definition of $\langle\langle A\rangle\rangle'$ by $\langle\langle A\rangle\rangle$.

$$\langle\langle A\rangle\rangle'\psi \leftrightarrow (\langle\langle A\rangle\rangle\psi \wedge \neg\langle\langle\emptyset\rangle\rangle\psi).$$

Actually $\langle\langle A\rangle\rangle$ and $\langle\langle A\rangle\rangle'$ are inter-definable and just differ in the choice of primitive.

$$\langle\langle A\rangle\rangle\psi \leftrightarrow (\langle\langle A\rangle\rangle'\psi \vee \langle\langle\emptyset\rangle\rangle\psi).$$

Positive condition is the core in the definition of $\langle\langle A\rangle\rangle'$ with an auxiliary negative condition. However, real world applications sometimes require both sides of control. A process may need to have the power of activating and inactivating another process, a memory allotment unit should be authorized in both access granting and access denying and so on. Such properties may be formalized by

$$\langle\langle A\rangle\rangle\psi \wedge \langle\langle A\rangle\rangle\neg\psi.$$

The sentence is not a well formed ATL formula since a non-temporal expression $\neg\psi$ immediately follows $\langle\langle A\rangle\rangle$. We therefore introduce a new operator $\langle\langle A\rangle\rangle_d$ to encapsulate both sides. We call $\langle\langle A\rangle\rangle_d$ *double control cooperation operators*. Sometime we also address it as indicating a full power of control.

> $q \models \langle\langle A\rangle\rangle_d\psi$, where ψ is a temporal formula, iff there exists collective strategies F_A and F'_A, one for each $a \in A$, such that $\lambda \models \psi$ for all computations $\lambda \in out(q, F_A)$, and $\lambda' \not\models \psi$ for all computations $\lambda' \in out(q, F'_A)$.

Players A in $\langle\langle A\rangle\rangle_d\psi$ have more influence on the value of ψ than that in $\langle\langle A\rangle\rangle'\psi$ and in $\langle\langle A\rangle\rangle\psi$. Formally we have validities

$$\langle\langle A\rangle\rangle_d\psi \rightarrow \langle\langle A\rangle\rangle'\psi.$$
$$\langle\langle A\rangle\rangle'\psi \rightarrow \langle\langle A\rangle\rangle\psi.$$

$\langle\langle\emptyset\rangle\rangle\psi$ is a satisfiable ATL formula. It states that empty set of players could have some functions. The problem is eliminated with $\langle\langle A\rangle\rangle_d$ t. Empty set is responsible for nothing, as shown in the following valid formula, where ψ is any temporal formula:

$$\neg\langle\langle\emptyset\rangle\rangle_d\psi.$$

Consider a simple example of two process system, say a and b. a assigns values to Boolean variable x. When $x = 0$, a leaves x unchanged or changes it to 1. When $x = 1$, a leaves x unchanged. Similarly, b assigns values to Boolean variable y. We model the synchronous composition of the two processes by the game structure $S = (\Sigma, Q, \Pi, \pi, \delta)$, where

$\Sigma = \{a, b\}$.
$Q = \{q, q_x, q_y, q_{xy}\}$.
$\Pi = \{x, y\}$.
$\pi(q) = \emptyset; \pi(q_x) = \{x\}; \pi(q_y) = \{y\}; \pi(q_{xy}) = \{x, y\}$.
$\delta(a, q) = \{\{q, q_y\}, \{q_x, q_{xy}\}\}; \delta(b, q) = \{\{q, q_x\}, \{q_y, q_{xy}\}\}$
$\delta(a, q_x) = \{\{q_x, q_{xy}\}\}; \delta(b, q_x) = \{\{q_x\}, \{q_{xy}\}\}$
$\delta(a, q_y) = \{\{q_y\}, \{q_{xy}\}\}; \delta(b, q_y) = \{\{q_y, q_{xy}\}\}$
$\delta(a, q_{xy}) = \{\{q_{xy}\}\}; \delta(b, q_{xy}) = \{\{q_{xy}\}\}$

First, clearly $S, q \models \langle\langle\{a\}\rangle\rangle \bigcirc (x \vee \neg x)$ because $x \vee \neg x$ is a tautology. However, $S, q \not\models \langle\langle\{a\}\rangle\rangle' \bigcirc (x \vee \neg x)$.

Second, we have $S, q \models \langle\langle\{a\}\rangle\rangle' \bigcirc (x \vee y)$ because $\bigcirc(x \vee y)$ is true at all computations where a changes the value of x to 1. $\bigcirc(x \vee y)$ is false at the computation where a and b leave both x and y unchanged. $S, q \not\models \langle\langle\{a\}\rangle\rangle_d \bigcirc (x \vee y)$ because there is no strategy for a to guarantee that $\bigcirc(x \vee y)$ is false. The value of $x \vee y$ depends on both a and b.

Third, $S, q \models \langle\langle\{a\}\rangle\rangle_d \bigcirc x$. a can either change the value of x to 1 so that $\bigcirc x$ is guaranteed to be true, or she can leave the value of x unchanged so that $\bigcirc x$ is guaranteed to be false. It is intuitive that a has an increasing power of control over $\bigcirc(x \vee \neg x)$, $\bigcirc(x \vee y)$, and $\bigcirc x$.

Proposition 3.1 $\langle\langle A\rangle\rangle_d$ is not definable in ATL.

Proof. The proof is sketched. Where p_1 and $p_2 \in \Pi$, we show that $\langle\langle A\rangle\rangle\neg(p_1 \mathcal{U} p_2)$ is not semantically equivalent to any ATL formula. Suppose otherwise. Let φ be its equivalent in ATL. Then $q \models \varphi$ iff there exist strategies F_A such that $\lambda \not\models p_1 \mathcal{U} p_2$ for all $\lambda \in out(q, F_A)$.

First, we observe that φ need not contain any operator $\langle\langle B\rangle\rangle$ with $B \neq A$. For any $B \neq A$ we can always construct a game structure such that for any A's strategy F_A and any B's strategy F_B, $out(q, F_A) \neq out(q, F_B)$. Therefore B is irrelevant in

evaluating φ.

Second, φ need not contain any Boolean combination of more than one formula of the form $\langle\langle A\rangle\rangle\psi$. For example, $\langle\langle A\rangle\rangle\psi_1 \wedge \langle\langle A\rangle\rangle\psi_2$. In general truth of $\langle\langle A\rangle\rangle\psi_1$ and truth of $\langle\langle A\rangle\rangle\psi_2$ depend on different strategies of A. Therefore we can always construct a game structure such that one of $\langle\langle A\rangle\rangle\psi_1$ and $\langle\langle A\rangle\rangle\psi_2$ is irrelevant in evaluating φ.

Third, φ need not contain any nested operator because $\langle\langle A\rangle\rangle\neg(p_1\mathcal{U}p_2)$ does not. A nested operator would subdivide an outcome, which is useless in evaluating φ.

We conclude that φ has a very simple form: it is a Boolean combination of propositions and a formula $\langle\langle A\rangle\rangle\psi$, where ψ can be either $\bigcirc\chi$, or $\Box\chi$, or $\chi\mathcal{U}\chi'$, where χ and χ' are propositional.

We already know from temporal logic studies that $\mathcal{U}$ is not definable by either $\bigcirc$ or $\Box$. Consider the case of $\chi\mathcal{U}\chi'$. Since $\langle\langle A\rangle\rangle\Box p$ is semantically equivalent to $\langle\langle A\rangle\rangle\neg(True\ \mathcal{U}\ \neg p)$, it is semantically equivalent to an ATL formula φ that is a Boolean combination of propositions and $\langle\langle A\rangle\rangle\chi\mathcal{U}\chi'$. When $A = \Sigma$, it is a definition of $E\Box$ from $E\mathcal{U}$. Contradict to the fact that $E\bigcirc$, $E\Box$, and $E\mathcal{U}$ is a minimal set of operators of CTL. QED

Conversely, it is clear that $\langle\langle A\rangle\rangle$ is not definable from $\langle\langle A\rangle\rangle_d$. Both operators should be primitive. From now on we use the name CATL (*causally alternating time temporal logic*) to denote the logic obtained by adding $\langle\langle A\rangle\rangle_d$ to ATL.

Note that unlike ATL, Boolean combination of temporal formulas is definable in CTL. For example, $E\neg(\varphi_1\mathcal{U}\varphi_2) =_{df} EG\neg\varphi_2 \vee E((\neg\varphi_2)\mathcal{U}(\neg\varphi_1 \wedge \neg\varphi_2))$.

Both $\langle\langle A\rangle\rangle$ and $\langle\langle A\rangle\rangle_d$ have monotonicity with respect to players.

Proposition 3.2 Where $A \subseteq B$, the following formulas are valid.

$\langle\langle A\rangle\rangle_d\psi \rightarrow \langle\langle B\rangle\rangle_d\psi$.
$\langle\langle A\rangle\rangle\psi \rightarrow \langle\langle B\rangle\rangle\psi$.

Proof. Suppose that $\langle\langle A\rangle\rangle_d\psi$ is true at q. That is, there exist collective strategies F_A and F'_A such that $\lambda \in out(q, F_A)$ implies $\lambda \models \psi$, and $\lambda' \in out(q, F'_A)$ implies $\lambda' \not\models \psi$. Let F_B (F'_B) be such that players in A take the same strategies as in F_A (F'_A). Then $out(q, F_B) \subseteq out(q, F_A)$ and $out(q, F'_B) \subseteq out(q, F'_A)$. Thus, $\lambda \in out(q, F_B)$ implies $\lambda \models \psi$, and $\lambda' \in out(q, F'_B)$ implies $\lambda' \not\models \psi$. QED

Exclusivity is a desired property of open systems. It states that a group of players A can guarantee a system entering into certain states, and no other group of players B can do the same unless A is included. Exclusivity indicates that A really functions as designed. Since in practice A is usually considered as one unit, we do not take into account cases of $A \cap B \neq \emptyset$. ATL does not have exclusivity. There exist A, B with $A \cap B = \emptyset$ such that both $\langle\langle A\rangle\rangle\psi$ and $\langle\langle B\rangle\rangle\psi$ are satisfiable.

Proposition 3.3 CATL has the property of exclusivity, that is, for any $A \cap B = \emptyset$ and any temporal formula ψ, we have a valid formula:

$$\langle\langle A\rangle\rangle_d\psi \rightarrow \neg\langle\langle B\rangle\rangle\psi.$$

Proof. Suppose that we have both $\langle\langle A\rangle\rangle_d\psi$ and $\langle\langle B\rangle\rangle\psi$. Then there exist collective strategies F_A and F_B such that $\lambda \in out(q, F_A)$ implies $\lambda \not\models \psi$, and $\lambda' \in out(q, F_B)$ implies $\lambda' \models \psi$. By definition 2.1 and $A \cap B = \emptyset$, $out(q, F_A) \cap out(q, F_B) \neq \emptyset$. Contradiction. QED

It is intuitive that full power of control implies exclusivity. However, the reverse is not true. The following statement does not hold in general.

If $q \models \langle\langle A\rangle\rangle\psi$ and $q \not\models \langle\langle B\rangle\rangle\psi$ for nay $A \cap B = \emptyset$, then $q \models \langle\langle A\rangle\rangle_d\psi$.

For a counterexample consider a two players game structure, say a and b. Let $c_1, c_2, c_3, c_4, c_5, c_6$ be the six computations passing through q. Let a have three choices $\{c_1, c_2\}$, $\{c_3, c_4\}$, and $\{c_5, c_6\}$, and b have two choices $\{c_1, c_3, c_5\}$, $\{c_2, c_4, c_6\}$. Let ψ be true at c_1, c_2, c_4, c_5, and false at c_3, c_6. It is easy to check that $q \models \langle\langle\{a\}\rangle\rangle\psi$ and $q \not\models \langle\langle\{b\}\rangle\rangle\psi$, but $q \not\models \langle\langle\{a\}\rangle\rangle_d\psi$. Thus full power of control is truly stronger than exclusivity.

For the last remark, $\langle\langle A\rangle\rangle_d$ is definable in a richer language ATL*. However, the model checking algorithm of ATL* is exponential in time [1]. A polynomial algorithm for CATL is present in the next section.

4. Strict alternating-time temporal logic

In the studies of *stit* theory on branching time, there is an idea called *strict stit* presented in [2]. Intuitively, for a set of players A, a formula of the form $[A\ stit : \varphi]$ means that players in A see to it that φ. Strict *stit* theory implements a more limited relation of "see to it that" by distinguishing the players in A according to their different roles in the action "see to it that φ". The idea is that a player in A, say a, is *essential* for the action in the sense that without a, the other players $A - \{a\}$ cannot see to it that φ; that is, $[A - \{a\}\ stit : \varphi]$ is false.

We introduce a similar idea into alternating-time temporal logic for our purposes.

Definition 4.1 *A set of players* A *is* quasi-essential *to a temporal formula φ at state* q *if* A *is a minimal set such that $q \models \langle\langle A\rangle\rangle\varphi$; that is, $q \models \langle\langle A\rangle\rangle\varphi$, but for any proper subset A' of* A, *$q \not\models \langle\langle A'\rangle\rangle\varphi$.*

That a set of players A is quasi-essential to φ at q means that players in A can cooperate to ensure that φ is true and that there is no redundant player in A with respect to this collective action. Note that this definition does not rule out that there

exists other set of players $A' \neq A$ such that A' is also quasi-essential to φ at q. That indicates that A is not essential for φ at q in the sense that more than one group of players can ensure the same outcome.

Definition 4.2 *A set of players* A *is* essential *to a temporal formula* φ *at state* q *if* $A = \bigcap\{B \mid B \text{ is quasi-essential to } \varphi \text{ at } q\}$; *For convenience, we also call each player* $a \in A$ essential *to* φ *at* q.

Suppose that in a system there are only three players a, b, and c, and we have $q \models \langle\langle\{a,b,c\}\rangle\rangle\varphi$, $q \models \langle\langle\{a,b\}\rangle\rangle\varphi$, and $q \models \langle\langle\{a,c\}\rangle\rangle\varphi$, and there is no other group of players can ensure φ at q. In this example, $\{a,b,c\}$ is not quasi-essential to φ at q, because its proper set $\{a,b\}$ (or $\{a,c\}$) can also ensure φ at q. $\{a,b\}$ and $\{a,c\}$ are quasi-essential to φ at q. However, neither of the two sets of players is essential. The only essential set is $\{a\}$.

A quasi-essential set represents a minimal sufficient condition for a temporal formula. An essential set represents a collection of necessary conditions for a temporal formula. Actually, it easy to see that an essential set collects all such necessary conditions.

The definition of "quasi-essential" shows which groups of players are needed to ensure a given outcome. And a given outcome can be implemented by different groups of players. Their intersection indicates the players indispensable to the outcome. Though there may be more than one quasi-essential group of players at a given situation, the essential group of players is always unique.

Consider a concurrent program designed for an authentication protocol as an example. A process that tackles information transmission is indispensable to the program. In our analysis this process is essential to the program specification. We may call such processes *core processes* (*core players*). In the above example, a process that tackles the system clock is not indispensable because it is possible to implement the program specification without an overall clock control: the synchronicity of the system can be implemented by other ways such as timestamps. Then a clock control process is not a core process and is not essential to the system specification. However, it is quasi-essential: It must be used if we implement synchronicity by a real clock control.

It is practically important to distinguish the quasi-essential or essential processes in a concurrent program. For quasi-essential processes, in a concurrent program, some processes may be redundant to a given task so that we do not need to consider them when we design or verify the system with respect to the task. This may reduce the cost of system design or specification. For essential processes, when there are several alternative groups of processes available to fulfill a given task and we want to make a choice among these alternatives, it is of benefit to know which processes are core processes. For, we only need to compare the non-core processes. We also can

expect that this will reduce the cost of system design and verification.

We extend ATL in order to express the above concepts. We call the resultant logic *strict alternating-time temporal logic* (SATL). The language of SATL consists of the language of ATL plus two new cooperation operators $\langle\langle A\rangle\rangle_{qe}$, read as "$A$ is quasi-essential to . . .", and $\langle\langle A\rangle\rangle_{e}$, read as "$A$ is essential to . . .." A formula of SATL is one of the following:

(S1) p, for propositions $p \in \Pi$.

(S2) $\neg\varphi$ or $\varphi_1 \vee \varphi_2$, where φ, φ_1 and φ_2 are SATL formulas.

(S3) $\langle\langle A\rangle\rangle\varphi$, $\langle\langle A\rangle\rangle_{qe}\varphi$, or $\langle\langle A\rangle\rangle_{e}\varphi$, where $A \subseteq \Sigma$ is a set of players, $\varphi \in \{\bigcirc\psi, \Box\psi, \psi_1 \mathcal{U} \psi_2\}$, and ψ, ψ_1, and ψ_2 are SATL formulas.

Similar to ATL, we write $\langle\langle A\rangle\rangle_{qe}\Diamond\varphi$ for $\langle\langle A\rangle\rangle_{qe} true\, \mathcal{U}\varphi$, and $\langle\langle A\rangle\rangle_{e}\Diamond\varphi$ for $\langle\langle A\rangle\rangle_{e} true\, \mathcal{U}\varphi$.

The semantics of SATL is based on a game structure $S = (\Sigma, Q, \Pi, \pi, \delta)$, as has defined. The semantic rules of satisfaction relation $\models$ for propositions, Boolean connectives, and cooperation operator $\langle\langle A\rangle\rangle$ are the same as ATL's. The rules for the two added operators are as follows:

$q \models \langle\langle A\rangle\rangle_{qe}\varphi$, where $\varphi \in \{\bigcirc\psi, \Box\psi, \varphi_1 \mathcal{U} \varphi_2\}$ iff $q \models \langle\langle A\rangle\rangle\varphi$, and there is no $A' \subset A$ such that $q \models \langle\langle A'\rangle\rangle\varphi$.

$q \models \langle\langle A\rangle\rangle_{e}\varphi$, where $\varphi \in \{\bigcirc\psi, \Box\psi, \varphi_1 \mathcal{U} \varphi_2\}$ iff $A = \bigcap\{B \mid q \models \langle\langle B\rangle\rangle_{qe}\varphi\}$.

Clearly, for any tautology φ, $\emptyset$ is the only quasi-essential set to φ at every state, and therefore $\emptyset$ is the essential set to φ at every state, which says that no player is essential to a tautology. The following lemma shows some facts about the new operators.

Lemma 4.3 *(1) If $q \models \langle\langle A\rangle\rangle\varphi$, then there is a $A' \subseteq A$ such that $q \models \langle\langle A'\rangle\rangle_{qe}\varphi$.*
(2) If $q \models \langle\langle A\rangle\rangle_{e}\varphi$, then for every set of players $B \neq A$, $q \not\models \langle\langle B\rangle\rangle_{e}\varphi$.

Consider the two players example as defined before. Clearly, $S, q \models \langle\langle\{a, b\}\rangle\rangle \bigcirc (x \vee y)$ because $\bigcirc(x \vee y)$ is true at all computations where player a (or b) takes a strategy such that when a (or b) is at state q, she changes the value of x (y) to 1. However, $\{a, b\}$ is not quasi-essential to $\bigcirc(x \vee y)$ at q because we have both $S, q \models \langle\langle\{a\}\rangle\rangle \bigcirc (x \vee y)$ and $S, q \models \langle\langle\{b\}\rangle\rangle \bigcirc (x \vee y)$, . It turns out that $S, q \not\models \langle\langle\{a, b\}\rangle\rangle_{qe} \bigcirc (x \vee y)$.

Actually, $\{a\}$ and $\{b\}$ are all quasi-essential sets to $\bigcirc(x \vee y)$ at q. Then $\emptyset$ is the essential set to $\bigcirc(x \vee y)$ at q and $S, q \models \langle\langle\emptyset\rangle\rangle_{qe} \bigcirc (x \vee y)$. This indicates that there is no essential player in this situation. Neither a nor b is core process. It is easy to see that $\{a\}$ is the only quasi-essential set to $\bigcirc x$ at q, and then is the essential set to $\bigcirc x$ at q. We have $S, q \models \langle\langle\{a\}\rangle\rangle_{e} \bigcirc x$.

5. Symbolic model checking

CATL model checking is the problem of finding the set of states in a game structure where a given CATL formula is satisfied. The algorithm is extended from the one in ([1]). It is implemented by a procedure $Check$ that takes a CATL formula φ as its argument and returns exactly the set of those states that satisfies φ. The output of $Check$ depends on the game structure being checked; this parameter is implicitly in the below algorithm. We define $Check$ inductively over the structure of CATL formulas. The cases for propositions and Boolean connectives are straightforward.

$Check(p) = Reg(p)$, where the function Reg returns the set of states that satisfy p.
$Check(\neg\varphi) = Check(True) - Check(\varphi)$.
$Check(\varphi_1 \vee \varphi_2) = Check(\varphi_1) \cup Check(\varphi_2)$.

The procedures for operator $\langle\langle A\rangle\rangle$ are from [1].

$Check(\langle\langle A\rangle\rangle \bigcirc \varphi) = Pre(A, Check(\varphi))$, where the function Pre returns the set of states q such that from q, the players in A can cooperate and enforce the next state to lie in $Check(\varphi)$.
$Check(\langle\langle A\rangle\rangle \Box\varphi)$ is implemented by the following procedure:

begin
$\rho := Check(True); \tau := Check(\varphi);$
while $\rho \not\subseteq \tau$ **do**
$\rho := \tau; \tau := Check(\varphi) \cap Pre(A, \rho)$
od;
return ρ.
end

$Check(\langle\langle A\rangle\rangle \varphi_1 \mathcal{U} \varphi_2)$ is implemented by the following procedure:

begin
$\rho := Check(False); \tau := Check(\varphi_2);$
while $\tau \not\subseteq \rho$ **do**
$\rho := \rho \cup \tau; \tau := Check(\varphi_1) \cap Pre(A, \rho)$
od;
return ρ.
end

For the operator $\langle\langle A\rangle\rangle_d$, we have that

$Check(\langle\langle A\rangle\rangle_d \bigcirc \varphi) = Check(\langle\langle A\rangle\rangle \bigcirc \varphi) \cap Check(\langle\langle A\rangle\rangle \bigcirc \neg\varphi)$;
$Check(\langle\langle A\rangle\rangle_d \Box\varphi) = Check(\langle\langle A\rangle\rangle \Box\varphi) \cap Check(\langle\langle A\rangle\rangle \Diamond\neg\varphi)$;
$Check(\langle\langle A\rangle\rangle_d \varphi_1 \mathcal{U} \varphi_2) = Check(\langle\langle A\rangle\rangle \varphi_1 \mathcal{U} \varphi_2) \cap NU(A, \varphi_1, \varphi_2)$, where $NU(A, \varphi_1, \varphi_2)$ is the following procedure:

begin
$\rho := Check(False); \tau := Check(\neg\varphi_1 \wedge \neg\varphi_2);$
while $\tau \not\subseteq \rho$ **do**
$\rho := \rho \cup \tau; \tau := Check(\varphi_1) \cap Pre(A, \rho)$
od;
return ρ.
end

To see how the procedure NU works, consider the result of the first circle: ρ is the set of states such that for each $q \in \rho$ implies $q \models (\neg\varphi_1 \wedge \neg\varphi_2) \vee (\varphi_1 \wedge \langle\langle A\rangle\rangle \bigcirc (\neg\varphi_1 \wedge \neg\varphi_2))$. ρ consists of states where A can guarantee $\varphi_1 \mathcal{U} \varphi_2$ being false, as long as paths whose length is no more than 2 are considered. The ending condition $\tau \subseteq \rho$ indicates that τ will not change in further circles, and therefore already contains all states we are searching for. Each circle in the procedure increases length of considered paths from n to $n + 1$.

Correctness of the algorithm can be proved by induction on the structure of an input formula. Termination is guaranteed, because the game structure being checked is finite.

It is easy to check that the complexity of CATL model checking is the same as that of ATL's. The last proposition is proved in the same way of theorem 5.2 in [1].

Proposition 5.1 *The model checking problem for CATL is PTIME-complete, and can be solved in time* $O(m \cdot l)$ *for a game structure with* m *transitions and an CATL formula of length* l.

For SATL model checking, it is the problem of finding the set of states in a game structure where a given SATL formula is satisfied. We also extend the ATL symbolic model checking algorithm for the purpose. The procedure $Check$ takes a SATL formula φ as its argument and returns exactly the set of those states that satisfies φ. $Check$ is defined inductively over the structure of SATL formulas. The cases for propositions, Boolean connectives, and cooperation operator $\langle\langle A\rangle\rangle$ are the same as before.

For the operator $\langle\langle A\rangle\rangle_{qe}$, let Sub be a function that, when given a set of players A, returns a queue of immediate subsets of A, that is, a queue of subsets of A with length $|A| - 1$. $Check(\langle\langle A\rangle\rangle_{qe}\varphi)$ is implemented by the following procedure:

begin
$\rho := Check(\langle\langle A\rangle\rangle\varphi)$
foreach $A' \in Sub(A)$ **do**
$\rho := \rho - Check(\langle\langle A'\rangle\rangle\varphi)$
od
return ρ.