Introduction to Propositional Dynamic Logic and Game Logic

Eric Pacuit

ILLC, University of Amsterdam

staff.science.uva.nl/~epacuit

epacuit@science.uva.nl

November 27, 2006

Introduction to Logic in Computer Science

Overview

- Proving Correctness of Programs: From Hoare Logic to **PDL**

- Introduction to Propositional Dynamic Logic (**PDL**)
- From **PDL** to Game Logic

- Semantics for Game Logic

- Example: Banach-Knaster Cake Cutting Procedure

What is a *Program*?

A computer program is a collection of instructions that describe a task, or set of tasks, to be carried out by a computer. (Wikepedia)

A *program* is a recipe written in a formal language for computing desired output data from given input data. (Harel, Kozen and Tiuryn)

Example: Euclid's Algorithm

```
x := u;
y := v;
while x ≠ y do
    if x < y then
        y := y − x;
    else
        x := x − y;
```

Input: $x, y \in \mathbb{N}$

Output: $gcd(x, y)$

When is a Program *Correct*?

Formal Specification use notations derived from formal logic to describe

- assumptions about the environment in which a program will operate
- requirements a program is to achieve
- how to design the program to achieve these goals

When is a Program *Correct*?

Formal Verification use methods of formal logic to

- validate specifications by checking consistency or posing challenges
- prove that a program satisfies the specification under given assumptions, or prove that a more detailed program implements a more abstract one.

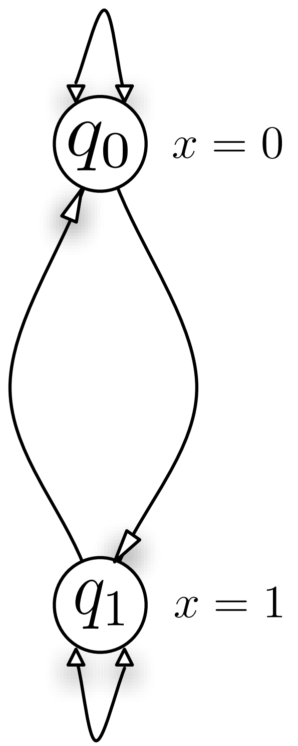
Exogenous and Endogenous Program Logics

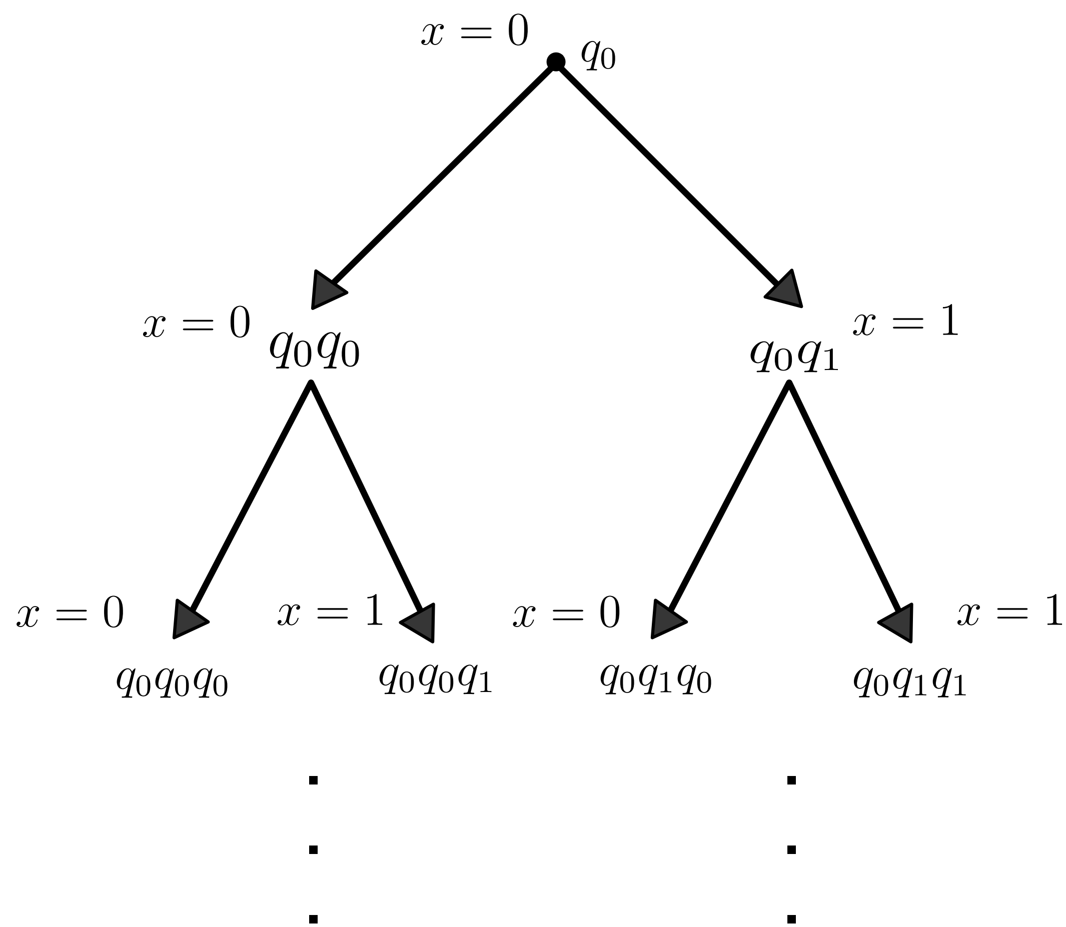
Two main approaches to the (modal) logic of programs:

Exogenous: programs are *explicit* in the formal language. Examples: Hoare Logic, Propositional Dynamic Logic (discussed today).

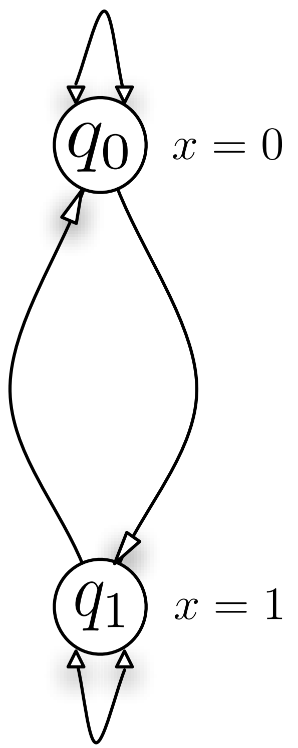
Endogenous: a program is fixed and considered part of the structure over which a program is interpreted. Examples: Linear and Branching Temporal Logics (not discussed today).

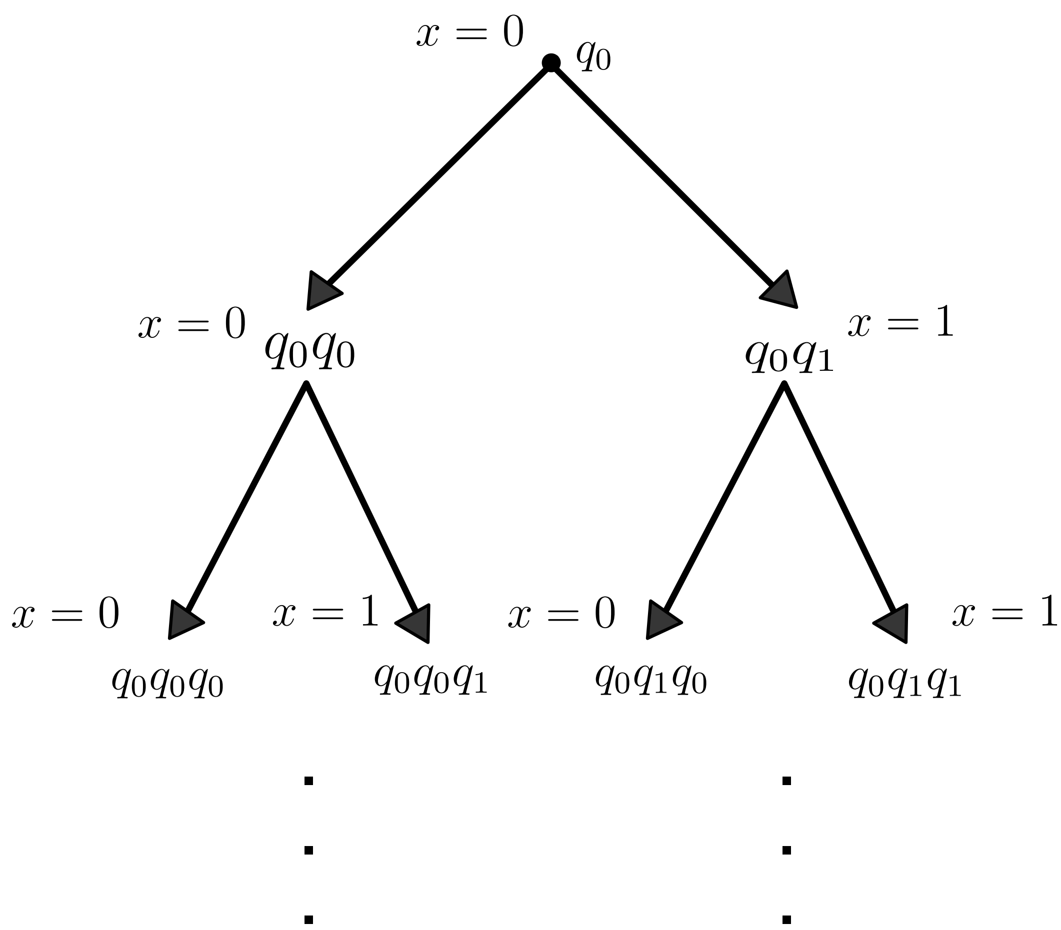
Computational vs. Behavioral Structures





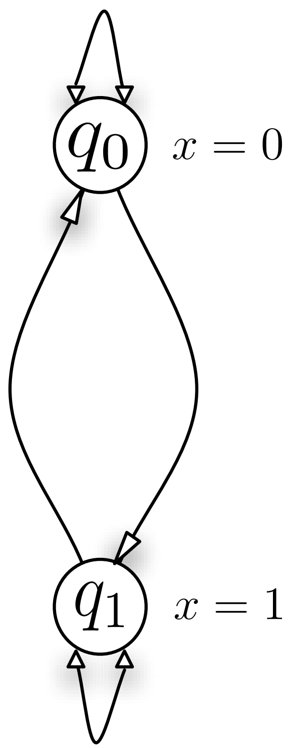
Computational vs. Behavioral Structures

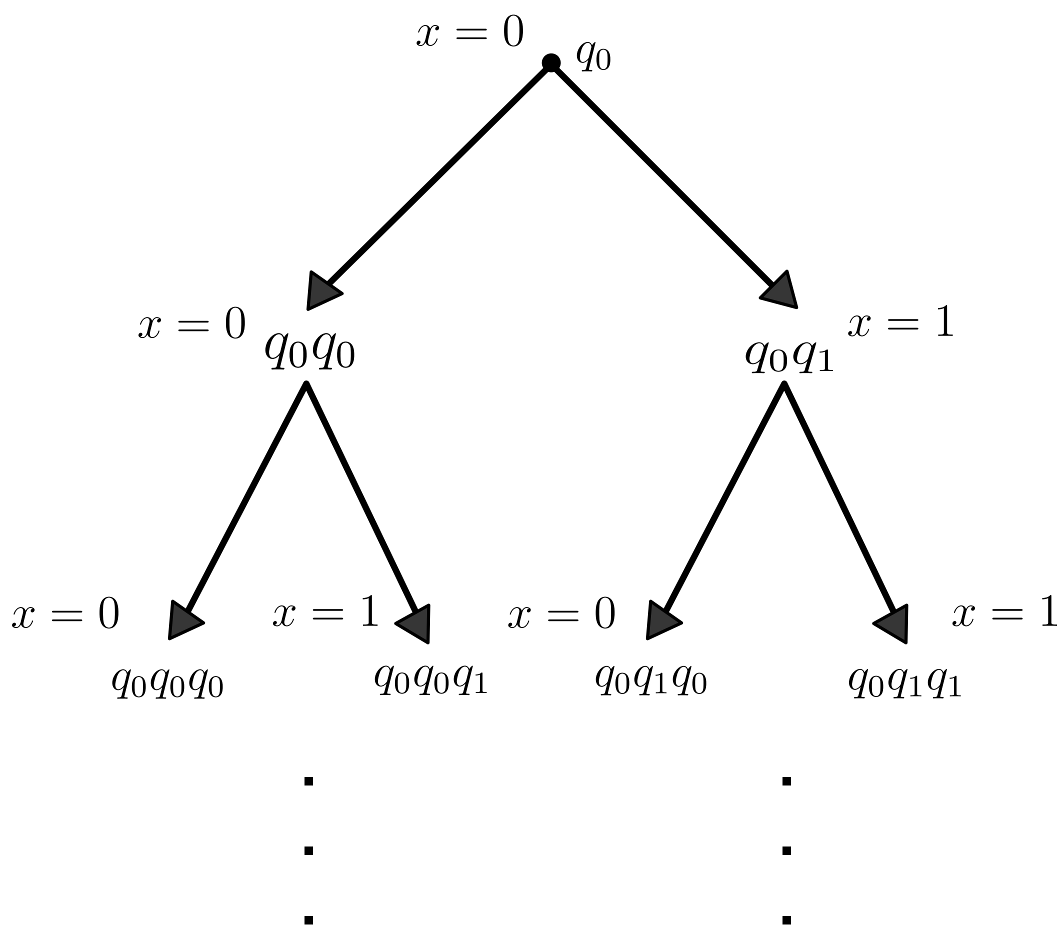




$\exists\Diamond P_{x=1}$

Computational vs. Behavioral Structures





$$\neg\forall\Diamond P_{x=1}$$

More on Temporal Logic

- *Linear Time Temporal Logic*: Reasoning about computation paths:

$$\Diamond\phi\text{: } \phi \text{ is true some time in } \textit{the} \text{ future.}$$

A. Pnuelli. *A Temporal Logic of Programs.* in *Proc. 18th IEEE Symposium on Foundations of Computer Science* (1977).

- *Branching Time Temporal Logic*: Allows quantification over paths:

$$\exists\Diamond\phi\text{: there is a path in which } \phi \text{ is eventually true.}$$

E. M. Clarke and E. A. Emerson. *Design and Synthesis of Synchronization Skeletons using Branching-time Temproal-logic Specifications.* In *Proceedings Workshop on Logic of Programs*, LNCS (1981).

Background: Hoare Logic

Background: Hoare Logic

Motivation: Formally verify the "correctness" of a program via *partial correctness assertions*:

$$\{\phi\}\alpha\{\psi\}$$

Background: Hoare Logic

Motivation: Formally verify the "correctness" of a program via *partial correctness assertions*:

$$\{\phi\}\alpha\{\psi\}$$

Intended Interpretation: If the program α begins in a state in which ϕ is true, then after α terminates (!), ψ will be true.

Background: Hoare Logic

Motivation: Formally verify the "correctness" of a program via *partial correctness assertions*:

$$\{\phi\}\alpha\{\psi\}$$

Intended Interpretation: If the program α begins in a state in which ϕ is true, then after α terminates (!), ψ will be true.

C. A. R. Hoare. *An Axiomatic Basis for Computer Programming.*. Comm. Assoc. Comput. Mach. 1969.

Background: Hoare Logic

Main Rules:

Background: Hoare Logic

Main Rules:

Assignment Rule: $\{\phi[x/e]\}\ x := e\ \{\phi\}$

Background: Hoare Logic

Main Rules:

Assignment Rule: $\{\phi[x/e]\}\ x := e\ \{\phi\}$

Composition Rule: $$\frac{\{\phi\}\ \alpha\ \{\sigma\} \qquad \{\sigma\}\ \beta\ \{\psi\}}{\{\phi\}\ \alpha;\beta\ \{\psi\}}$$

Background: Hoare Logic

Main Rules:

Assignment Rule: $\{\phi[x/e]\}\ x := e\ \{\phi\}$

Composition Rule: $$\frac{\{\phi\}\ \alpha\ \{\sigma\} \qquad \{\sigma\}\ \beta\ \{\psi\}}{\{\phi\}\ \alpha; \beta\ \{\psi\}}$$

Conditional Rule: $$\frac{\{\phi \wedge \sigma\}\ \alpha\ \{\psi\} \qquad \{\phi \wedge \neg\sigma\}\ \beta\ \{\psi\}}{\{\phi\}\ \textbf{if}\ \sigma\ \textbf{then}\ \alpha\ \textbf{else}\ \beta\ \{\psi\}}$$

Background: Hoare Logic

Main Rules:

Assignment Rule: $\{\phi[x/e]\}\ x := e\ \{\phi\}$

Composition Rule: $$\frac{\{\phi\}\ \alpha\ \{\sigma\} \qquad \{\sigma\}\ \beta\ \{\psi\}}{\{\phi\}\ \alpha;\beta\ \{\psi\}}$$

Conditional Rule: $$\frac{\{\phi \land \sigma\}\ \alpha\ \{\psi\} \qquad \{\phi \land \neg\sigma\}\ \beta\ \{\psi\}}{\{\phi\}\ \textbf{if}\ \sigma\ \textbf{then}\ \alpha\ \textbf{else}\ \beta\ \{\psi\}}$$

While Rule: $$\frac{\{\phi \land \sigma\}\ \alpha\ \{\phi\}}{\{\phi\}\ \textbf{while}\ \sigma\ \textbf{do}\ \alpha\ \{\phi \land \neg\sigma\}}$$

Example: Euclid's Algorithm

```
x := u;
y := v;
while x ≠ y do
    if x < y then
        y := y − x;
    else
        x := x − y;
```

Let $\phi := \gcd(x, y) = \gcd(u, v)$

Example: Euclid's Algorithm

```
x := u;
y := v;
while x ≠ y do
    if x < y then
        y := y − x;
    else
        x := x − y;
```

Let α be the inner `if` statement.

Example: Euclid's Algorithm

```
x := u;
y := v;
while x ≠ y do
   if x < y then
      y := y − x;
   else
      x := x − y;
```

Let α be the inner `if` statement.

Then $\{\gcd(x, y) = \gcd(u, v)\}\ \alpha\ \{\gcd(x, y) = \gcd(u, v)\}$

Example: Euclid's Algorithm

```
x := u;
y := v;
while x ≠ y do
   if x < y then
      y := y − x;
   else
      x := x − y;
```

Hence by the `while`-rule (using a "weakening rule")

$$\frac{\{(\gcd(x,y)=\gcd(u,v))\wedge(x\neq y)\}\ \alpha\ \{\gcd(x,y)=\gcd(u,v))\}}{\{\gcd(x,y)=\gcd(u,v)\}\ \textbf{while}\ \sigma\ \textbf{do}\ \alpha\ \{(\gcd(x,y)=\gcd(u,v))\wedge\neg(x\neq y)\}}$$

More on Hoare Logic

K. Apt. *Ten Years of Hoare's Logic: A Survey — Part I.* ACM Transactions on Programming Languages and Systems, 1981.

Programs as State Transformers

A *state* is (informally) an instantaneous description of reality.

Formally, a state of a program is a function that assigns to every variable a value from its domain of interpretation.

Example: $(x, y, u, v) = (15, 27, 15, 27)$ is the initial state of the above program.

Programs as State Transformers

A *state* is (informally) an instantaneous description of reality.

Formally, a state of a program is a function that assigns to every variable a value from its domain of interpretation.

Example: $(x, y, u, v) = (15, 27, 15, 27)$ is the initial state of the above program.

A program is a set of instructions that *transforms a state*:

$$(15, 27, 15, 27), (15, 12, 15, 27), (3, 12, 15, 27),$$
$$(3, 9, 15, 27), (3, 6, 15, 27), (3, 3, 15, 27)$$

Propositional Dynamic Logic

Let P be a set of atomic programs and At a set of atomic propositions.

Formulas of **PDL** have the following syntactic form:

$$\phi := p \mid \bot \mid \neg\phi \mid \phi \vee \psi \mid [\alpha]\phi$$

$$\alpha := a \mid \alpha \cup \beta \mid \alpha;\beta \mid \alpha^{*} \mid \phi?$$

where $p \in \mathsf{At}$ and $a \in \mathsf{P}$.

Propositional Dynamic Logic

Let P be a set of atomic programs and At a set of atomic propositions.

Formulas of **PDL** have the following syntactic form:

$$\phi := p \mid \bot \mid \neg\phi \mid \phi \vee \psi \mid [\alpha]\phi$$

$$\alpha := a \mid \alpha \cup \beta \mid \alpha;\beta \mid \alpha^* \mid \phi?$$

where $p \in \mathsf{At}$ and $a \in \mathsf{P}$.

$\{\phi\}\ \alpha\ \{\psi\}$ **is replaced with** $\phi \rightarrow [\alpha]\psi$

PDL: Intended Meanings

$[\alpha]\phi$: "It is necessary that after executing α, ϕ is true"

$\alpha;\beta$: "Execute α then β

$\alpha \cup \beta$: "Choose either α or β

α^*: "Execute α a nondeterministically finite number of times (zero or more)"

$\phi?$: "Test ϕ, proceed if true, fail if false"

Familiar Programs

$$
\begin{aligned}
\textbf{skip} &:= \top?\\
\textbf{fail} &:= \bot?\\
\textbf{if } \phi \textbf{ then } \alpha \textbf{ else } \beta &:= \phi?;\alpha \cup \neg\phi?;\beta\\
\textbf{while } \phi \textbf{ do } \alpha &:= (\phi?;\alpha)^*;\neg\phi?\\
\textbf{do } \phi_1 \rightarrow \alpha_1 \mid \cdots \mid \phi_n \rightarrow \alpha_n \textbf{ od} &:= (\phi_1?;\alpha_1 \cup \cdots \cup \phi_n?;\alpha_n)^*;\\
& (\neg\phi_1 \wedge \cdots \wedge \neg\phi_n)?
\end{aligned}
$$

Semantics

Semantics: $\mathcal{M} = \langle W, \{R_a \mid a \in \mathsf{P}\}, V\rangle$ where for each $a \in \mathsf{P}$, $R_a \subseteq W \times W$ and $V : \mathsf{At} \to 2^W$

- $R_{\alpha\cup\beta} := R_\alpha \cup R_\beta$
- $R_{\alpha;\beta} := R_\alpha \circ R_\beta$
- $R_{\alpha^*} := \cup_{n\geq 0} R_\alpha^n$
- $R_{\phi?} = \{(w, w) \mid \mathcal{M}, w \models \phi\}$

$$\mathcal{M}, w \models [\alpha]\phi \text{ iff for each } v, \text{ if } wR_\alpha v \text{ then } \mathcal{M}, v \models \phi$$

Segerberg Axioms

1. Axioms of propositional logic
2. $[\alpha](\phi \rightarrow \psi) \rightarrow ([\alpha]\phi \rightarrow [\alpha]\psi)$
3. $[\alpha \cup \beta]\phi \leftrightarrow [\alpha]\phi \wedge [\beta]\phi$
4. $[\alpha; \beta]\phi \leftrightarrow [\alpha][\beta]\phi$
5. $[\psi?]\phi \leftrightarrow (\psi \rightarrow \phi)$
6. $\phi \wedge [\alpha][\alpha^*]\phi \leftrightarrow [\alpha^*]\phi$
7. $\phi \wedge [\alpha^*](\phi \rightarrow [\alpha]\phi) \rightarrow [\alpha^*]\phi$
8. Modus Ponens and Necessitation (for each program α)

Segerberg Axioms

1. Axioms of propositional logic
2. $[\alpha](\phi \rightarrow \psi) \rightarrow ([\alpha]\phi \rightarrow [\alpha]\psi)$
3. $[\alpha \cup \beta]\phi \leftrightarrow [\alpha]\phi \wedge [\beta]\phi$
4. $[\alpha ; \beta]\phi \leftrightarrow [\alpha][\beta]\phi$
5. $[\psi ?]\phi \leftrightarrow (\psi \rightarrow \phi)$
6. $\phi \wedge [\alpha][\alpha^*]\phi \leftrightarrow [\alpha^*]\phi$ (Fixed-Point Axiom)
7. $\phi \wedge [\alpha^*](\phi \rightarrow [\alpha]\phi) \rightarrow [\alpha^*]\phi$ (Induction Axiom)
8. Modus Ponens and Necessitation (for each program α)

Extending PDL

- Intersection ($\alpha \cap \beta$): $wR_{\alpha\cap\beta}v$ iff $wR_\alpha v$ and $wR_\beta v$
- Complementation ($\overline{\alpha}$): $wR_{\overline{\alpha}}v$ iff it is not the case that $wR_\alpha v$
- Converse (α^{-1}): $wR_{\alpha^{-1}}v$ iff $vR_\alpha w$

Extending PDL

- Intersection ($\alpha \cap \beta$): $wR_{\alpha\cap\beta}v$ iff $wR_\alpha v$ and $wR_\beta v$
- Complementation ($\overline{\alpha}$): $wR_{\overline{\alpha}}v$ iff it is not the case that $wR_\alpha v$
- Converse (α^{-1}): $wR_{\alpha^{-1}}v$ iff $vR_\alpha w$

See work on *Boolean modal logic*.

Passy and Tinchev. *An essay in combinatory dynamic logic*. Information and Computation **93** (1991).

Extending PDL

- Intersection ($\alpha \cap \beta$): $wR_{\alpha\cap\beta}v$ iff $wR_\alpha v$ and $wR_\beta v$
- Complementation ($\overline{\alpha}$): $wR_{\overline{\alpha}}v$ iff it is not the case that $wR_\alpha v$
- Converse (α^{-1}): $wR_{\alpha^{-1}}v$ iff $vR_\alpha w$

See work on *Boolean modal logic*.

Passy and Tinchev. *An essay in combinatory dynamic logic*. Information and Computation **93** (1991).

Dynamic Logic (first-order), nonregular PDL, etc.

D. Harel, D. Kozen and Tiuryn. *Dynamic Logic*. 2000.

A Survey of Results

Theorem (Parikh, Kozen and Parikh) **PDL** is sound and *weakly* complete with respect to the Segerberg Axioms.

Theorem The satisfiability problem for **PDL** is decidable (EXPTIME-Complete).

D. Kozen and R. Parikh. *An Elementary Proof of the Completeness of PDL.* 1981.

D. Harel, D. Kozen and Tiuryn. *Dynamic Logic.* 2000.

A Survey of Results

R is deterministic if for each $w \in W$ there is a unique $v \in W$ such that wRv.

Theorem Assuming that all atomic programs are deterministic, then PDL with intersection is highly undecidable, i.e., Π^1_1-complete.

A Survey of Results

R is deterministic if for each $w \in W$ there is a unique $v \in W$ such that wRv.

Theorem Assuming that all atomic programs are deterministic, then PDL with intersection is highly undecidable, i.e., Π_1^1-complete.

Theorem The satisfiability problem for PDL with complementation is undecidable.

A Survey of Results

R is deterministic if for each $w \in W$ there is a unique $v \in W$ such that wRv.

Theorem Assuming that all atomic programs are deterministic, then PDL with intersection is highly undecidable, i.e., Π^1_1-complete.

Theorem The satisfiability problem for PDL with complementation is undecidable.

Let S_ϕ be all the substitution instances of ϕ.

Theorem The problem of deciding whether $S_\phi \models \psi$ is *highly undecidable* (Π^1_1-complete).

D. Harel, D. Kozen and Tiuryn. *Dynamic Logic*. 2000.

A Survey of (Recent) Results

Theorem PDL with intersection but without iteration is axiomatizable.

Balbiani and Vakarelov. *Iteration-free pdl with intersection: a complete axiomatization.* Fundamenta Informaticae **45** (2001) .

Theorem The satisfiability problem for PDL with complement applied only to atomic programs is decidable.

Lutz and Walther. *PDL with negation of atomic programs.* Journal of Applied Non-Classical Logics **14(2)** (2004) .

A Survey of (Recent) Results

Theorem The satisfiability problem for PDL with intersection (and without complement) is 2-EXPSPACE-complete.

Lange and Lutz. *2-exptime lower bounds for propositional dynamic logics with intersection.* Journal of Symbolic Logic **70(4)** (2005).

Theorem The model checking problem for PDL models is PTIME.

Lange. *Model checking propositional dynamic logic with all extras.* Journal of Applied Logic **4** (2006).

From PDL to Game Logic

Game Logic (GL) was introduced by Rohit Parikh in

R. Parikh. *The Logic of Games and its Applications.*. Annals of Discrete Mathematics. (1985) .

From PDL to Game Logic

Game Logic (**GL**) was introduced by Rohit Parikh in

R. Parikh. *The Logic of Games and its Applications.*. Annals of Discrete Mathematics. (1985) .

Main Idea:

In **PDL**: $w \models \langle\pi\rangle\phi$: there is a run of the program π starting in state w that ends in a state where ϕ is true.

The programs in **PDL** can be thought of as *single player games*.

From PDL to Game Logic

Game Logic (**GL**) was introduced by Rohit Parikh in

R. Parikh. *The Logic of Games and its Applications.*. Annals of Discrete Mathematics. (1985) .

Main Idea:

In **PDL**: $w \models \langle\pi\rangle\phi$: there is a run of the program π starting in state w that ends in a state where ϕ is true.

The programs in **PDL** can be thought of as *single player games*.

Game Logic generalized **PDL** by considering two players:

In **GL**: $w \models \langle\gamma\rangle\phi$: Angel has a **strategy** in the game γ to ensure that the game ends in a state where ϕ is true.

From PDL to Game Logic

Consequences of two players:

From PDL to Game Logic

Consequences of two players:

$\langle\gamma\rangle\phi$: Angel has a strategy in γ to ensure ϕ is true

$[\gamma]\phi$: Demon has a strategy in γ to ensure ϕ is true

From PDL to Game Logic

Consequences of two players:

$\langle\gamma\rangle\phi$: Angel has a strategy in γ to ensure ϕ is true

$[\gamma]\phi$: Demon has a strategy in γ to ensure ϕ is true

Either Angel or Demon can win: $\langle\gamma\rangle\phi \vee [\gamma]\neg\phi$

From PDL to Game Logic

Consequences of two players:

$\langle\gamma\rangle\phi$: Angel has a strategy in γ to ensure ϕ is true

$[\gamma]\phi$: Demon has a strategy in γ to ensure ϕ is true

Either Angel or Demon can win: $\langle\gamma\rangle\phi \vee [\gamma]\neg\phi$

But not both: $\neg(\langle\gamma\rangle\phi \wedge [\gamma]\neg\phi)$

From PDL to Game Logic

Consequences of two players:

$\langle\gamma\rangle\phi$: Angel has a strategy in γ to ensure ϕ is true

$[\gamma]\phi$: Demon has a strategy in γ to ensure ϕ is true

Either Angel or Demon can win: $\langle\gamma\rangle\phi \vee [\gamma]\neg\phi$

But not both: $\neg(\langle\gamma\rangle\phi \wedge [\gamma]\neg\phi)$

Thus, $[\gamma]\phi \leftrightarrow \neg\langle\gamma\rangle\neg\phi$ is a valid principle

From PDL to Game Logic

Consequences of two players:

$\langle\gamma\rangle\phi$: Angel has a strategy in γ to ensure ϕ is true

$[\gamma]\phi$: Demon has a strategy in γ to ensure ϕ is true

Either Angel or Demon can win: $\langle\gamma\rangle\phi \vee [\gamma]\neg\phi$

But not both: $\neg(\langle\gamma\rangle\phi \wedge [\gamma]\neg\phi)$

Thus, $[\gamma]\phi \leftrightarrow \neg\langle\gamma\rangle\neg\phi$ is a valid principle

However, $[\gamma]\phi \wedge [\gamma]\psi \rightarrow [\gamma](\phi \wedge \psi)$ is **not** a valid principle

From PDL to Game Logic

Reinterpret operations and invent new ones:

- $?\phi$: Check whether ϕ currently holds

From PDL to Game Logic

Reinterpret operations and invent new ones:

- $?\phi$: Check whether ϕ currently holds
- $\gamma_1;\gamma_2$: First play γ_1 then γ_2

From PDL to Game Logic

Reinterpret operations and invent new ones:

- $?\phi$: Check whether ϕ currently holds
- $\gamma_1 ; \gamma_2$: First play γ_1 then γ_2
- $\gamma_1 \cup \gamma_2$: Angel choose between γ_1 and γ_2

From PDL to Game Logic

Reinterpret operations and invent new ones:

- $?\phi$: Check whether ϕ currently holds
- $\gamma_1;\gamma_2$: First play γ_1 then γ_2
- $\gamma_1 \cup \gamma_2$: Angel choose between γ_1 and γ_2
- γ^*: Angel can choose how often to play γ (possibly not at all); each time she has played γ, she can decide whether to play it again or not.

From PDL to Game Logic

Reinterpret operations and invent new ones:

- $?\phi$: Check whether ϕ currently holds
- $\gamma_1;\gamma_2$: First play γ_1 then γ_2
- $\gamma_1 \cup \gamma_2$: Angel choose between γ_1 and γ_2
- γ^*: Angel can choose how often to play γ (possibly not at all); each time she has played γ, she can decide whether to play it again or not.
- γ^d: Switch roles, then play γ

From PDL to Game Logic

Reinterpret operations and invent new ones:

- $?\phi$: Check whether ϕ currently holds
- $\gamma_1;\gamma_2$: First play γ_1 then γ_2
- $\gamma_1 \cup \gamma_2$: Angel choose between γ_1 and γ_2
- γ^*: Angel can choose how often to play γ (possibly not at all); each time she has played γ, she can decide whether to play it again or not.
- γ^d: Switch roles, then play γ
- $\gamma_1 \cap \gamma_2 := (\gamma_1^d \cup \gamma_2^d)^d$: Demon chooses between γ_1 and γ_2

From PDL to Game Logic

Reinterpret operations and invent new ones:

- $?\phi$: Check whether ϕ currently holds
- $\gamma_1;\gamma_2$: First play γ_1 then γ_2
- $\gamma_1 \cup \gamma_2$: Angel choose between γ_1 and γ_2
- γ^*: Angel can choose how often to play γ (possibly not at all); each time she has played γ, she can decide whether to play it again or not.
- γ^d: Switch roles, then play γ
- $\gamma_1 \cap \gamma_2 := (\gamma_1^d \cup \gamma_2^d)^d$: Demon chooses between γ_1 and γ_2
- $\gamma^x := ((\gamma^d)^*)^d$: Demon can choose how often to play γ (possibly not at all); each time he has played γ, he can decide whether to play it again or not.

Syntax

Let Γ_0 be a set of atomic games and At a set of atomic propositions. Then formulas of Game Logic are defined inductively as follows:

$$\gamma \ := \ g \mid \phi? \mid \gamma;\gamma \mid \gamma \cup \gamma \mid \gamma^* \mid \gamma^d$$

$$\phi \ := \ \bot \mid p \mid \neg\phi \mid \phi \vee \phi \mid \langle\gamma\rangle\phi \mid [\gamma]\phi$$

where $p \in \mathsf{At}, g \in \Gamma_0$.

Game Logic: Semantics I

A **neighborhood game model** is a tuple
$\mathcal{M} = \langle W, \{E_g \mid g \in \Gamma_0\}, V\rangle$ where

W is a nonempty set of states

For each $g \in \Gamma_0$, $E_g : W \rightarrow 2^{2^W}$ is an **effectivity function** such that if $X \subseteq X'$ and $X \in E_g(w)$ then $X' \in E_g(w)$.

$X \in E_g(w)$ means in state s, Angel has a strategy to force the game to end in *some* state in X (we may write wE_gX)

$V : \mathsf{At} \rightarrow 2^W$ is a valuation function.

Game Logic: Semantics I

A **neighborhood game model** is a tuple
$\mathcal{M} = \langle W, \{E_g \mid g \in \Gamma_0\}, V \rangle$ where

Propositional letters and boolean connectives are as usual.

$\mathcal{M}, w \models \langle \gamma \rangle \phi$ iff $(\phi)^{\mathcal{M}} \in E_\gamma(w)$

Game Logic: Semantics I

A **neighborhood game model** is a tuple
$\mathcal{M} = \langle W, \{E_g \mid g \in \Gamma_0\}, V\rangle$ where

Propositional letters and boolean connectives are as usual.

$\mathcal{M}, w \models \langle\gamma\rangle\phi$ iff $(\phi)^{\mathcal{M}} \in E_\gamma(w)$

Suppose $E_\gamma(Y) = \{s \mid Y \in E_g(s)\}$

- $E_{\gamma_1;\gamma_2}(Y) \;:=\; E_{\gamma_1}(E_{\gamma_2}(Y))$
- $E_{\gamma_1\cup\gamma_2}(Y) \;:=\; E_{\gamma_1}(Y) \cup E_{\gamma_2}(Y)$
- $E_{\phi?}(Y) \;:=\; (\phi)^{\mathcal{M}} \cap Y$
- $E_{\gamma^d}(Y) \;:=\; \overline{E_\gamma(\overline{Y})}$
- $E_{\gamma^*}(Y) \;:=\; \mu X.Y \cup E_\gamma(X)$

Game Logic: Axioms

1. All propositional tautologies
2. $\langle \alpha ; \beta \rangle \phi \leftrightarrow \langle \alpha \rangle \langle \beta \rangle \phi$ Composition
3. $\langle \alpha \cup \beta \rangle \phi \leftrightarrow \langle \alpha \rangle \phi \vee \langle \beta \rangle \phi$ Union
4. $\langle \psi ? \rangle \phi \leftrightarrow (\psi \wedge \phi)$ Test
5. $\langle \alpha^d \rangle \phi \leftrightarrow \neg \langle \alpha \rangle \neg \phi$ Dual
6. $(\phi \vee \langle \alpha \rangle \langle \alpha^* \rangle \phi) \rightarrow \langle \alpha^* \rangle \phi$ Mix

and the rules,

$$\frac{\phi \qquad \phi \rightarrow \psi}{\psi} \qquad \frac{\phi \rightarrow \psi}{\langle \alpha \rangle \phi \rightarrow \langle \alpha \rangle \psi} \qquad \frac{(\phi \vee \langle \alpha \rangle \psi) \rightarrow \psi}{\langle \alpha^* \rangle \phi \rightarrow \psi}$$

Some Results

- Game Logic is more expressive than **PDL**

Some Results

- Game Logic is more expressive than **PDL**

$$\langle (g^d)^* \rangle \bot$$

Some Results

- Game Logic is more expressive than **PDL**

$$\langle (g^d)^* \rangle \bot$$

- The induction axiom is not valid in GL.

R. Parikh. *The Logic of Games and its Applications.*. Annals of Discrete Mathematics. (1985) .

Some Results

- Game Logic is more expressive than **PDL**

$$\langle (g^d)^* \rangle \bot$$

- The induction axiom is not valid in GL.

R. Parikh. *The Logic of Games and its Applications.*. Annals of Discrete Mathematics. (1985) .

- All GL games are determined. This is not a trivial result since neither Zermelo's Theorem nor the Gale-Stewart Theorem can be applied.

M. Pauly. *Game Logic for Game Theorists.* Available at `http://www.stanford.edu/ pianoman/`.

Some Results

Theorem [1] Dual-free game logic is sound and complete with respect to the class of all game models.

Theorem [2] Iteration-free game logic is sound and complete with respect to the class of all game models.

Open Question Is (full) game logic complete with respect to the class of all game models?

[1] R. Parikh. *The Logic of Games and its Applications.*. Annals of Discrete Mathematics. (1985) .

[2] M. Pauly. *Logic for Social Software.* Ph.D. Thesis, University of Amsterdam (2001)..

Some Results

Theorem [2] Given a game logic formula ϕ and a finite game model $\mathcal{M}$, model checking can be done in time $O(|\mathcal{M}|^{ad(\phi)+1} \times |\phi|)$

Theorem [1,2] The satisfiability problem for game logic is in EXPTIME.

Theorem [1] Game logic can be translated into the modal μ-calculus

[1] R. Parikh. *The Logic of Games and its Applications.*. Annals of Discrete Mathematics. (1985) .

[2] M. Pauly. *Logic for Social Software.* Ph.D. Thesis, University of Amsterdam (2001)..

Some Results

Say two games γ_1 and γ_2 are equivalent provided $E_{\gamma_1} = E_{\gamma_2}$ iff $\langle\gamma_1\rangle p \leftrightarrow \langle\gamma_2\rangle p$ is valid for a p which occurs neither in γ_1 nor in γ_2.

Theorem [1,2]] Sound and complete axiomatizations of (iteration free) game logic

Theorem [3] No finite level of the modal μ-calculus hierarchy captures the expressive power of game logic.

[1] Y. Venema. *Representing Game Algebras.* Studia Logica **75** (2003)..

[2] V. Goranko. *The Basic Algebra of Game Equivalences.* Studia Logica **75** (2003)..

[3] D. Berwanger. *Game Logic is Strong Enough for Parity Games.* Studia Logica **75** (2003)..

More Information

Editors: M. Pauly and R. Parikh. *Special Issue on Game Logic*. Studia Logica **75**, 2003.

M. Pauly and R. Parikh. *Game Logic — An Overview*. Studia Logica **75**, 2003.

R. Parikh. *The Logic of Games and its Applications.*. Annals of Discrete Mathematics. (1985) .

M. Pauly. *Game Logic for Game Theorists*. Available at `http://www.stanford.edu/ pianoman/`.

Example: Banach-Knaster Cake Cutting Algorithm

The Algorithm:

Example: Banach-Knaster Cake Cutting Algorithm

The Algorithm:

- The first person cuts out a piece which he claims is his fair share.

Example: Banach-Knaster Cake Cutting Algorithm

The Algorithm:

- The first person cuts out a piece which he claims is his fair share.
- The piece goes around being inspected by each agent.

Example: Banach-Knaster Cake Cutting Algorithm

The Algorithm:

- The first person cuts out a piece which he claims is his fair share.
- The piece goes around being inspected by each agent.
- Each agent, in turn, can either reduce the piece, putting some back to the main part, or just pass it.

Example: Banach-Knaster Cake Cutting Algorithm

The Algorithm:

- The first person cuts out a piece which he claims is his fair share.
- The piece goes around being inspected by each agent.
- Each agent, in turn, can either reduce the piece, putting some back to the main part, or just pass it.
- After the piece has been inspected by p_n, the last person who reduced the piece, takes it. If there is no such person, then the piece is taken by p_1.

Example: Banach-Knaster Cake Cutting Algorithm

The Algorithm:

- The first person cuts out a piece which he claims is his fair share.
- The piece goes around being inspected by each agent.
- Each agent, in turn, can either reduce the piece, putting some back to the main part, or just pass it.
- After the piece has been inspected by p_n, the last person who reduced the piece, takes it. If there is no such person, then the piece is taken by p_1.
- The algorithm continues with $n-1$ participants.

Example: Banach-Knaster Cake Cutting Algorithm

Correctness: The algorithm is "correct" iff each player has a winning strategy for achieving a fair outcome ($1/n$ of the pie according to p_i's own valuation).

Towards a Formal Proof: A state will consist of the values of $n+2$ variables.

- The variable m has as its value the main part of the cake.
- The variable x is the piece under consideration.
- For $i = 1, \ldots, n$, the variable x_i has as its value the piece, if any, assigned to the person p_i.

Variables $m, x, x_1, \ldots, x_n$ range over subsets of the cake.

Example: Banach-Knaster Cake Cutting Algorithm

The algorithm uses three basic actions.

- c cuts a piece from m and assigns it to x. c works only if x is 0.
- r (reduce) transfers some (non-zero) portion from x back to m.
- a_i (assign) assigns the piece x to person p_i. Thus a_i is simply, $(x_i, x) := (x, 0)$.

Example: Banach-Knaster Cake Cutting Algorithm

The algorithm uses three basic actions.

- c cuts a piece from m and assigns it to x. c works only if x is 0.
- r (reduce) transfers some (non-zero) portion from x back to m.
- a_i (assign) assigns the piece x to person p_i. Thus a_i is simply, $(x_i, x) := (x, 0)$.

And predicates:

- $F(u, k)$: the piece u is big enough for k people.
- $F(u)$ abbreviates $F(u, 1)$ and F_i abbreviates $F(x_i)$.

Example: Banach-Knaster Cake Cutting Algorithm

Assume the following propositions

Example: Banach-Knaster Cake Cutting Algorithm

Assume the following propositions

1. $F(m, k) \rightarrow \langle c \rangle (F(m, k-1) \wedge F(x))$

Example: Banach-Knaster Cake Cutting Algorithm

Assume the following propositions

1. $F(m,k) \rightarrow \langle c \rangle (F(m,k-1) \wedge F(x))$

1′. $F(m,k) \rightarrow (c,i)(F(m,k-1) \wedge F(x))$

Example: Banach-Knaster Cake Cutting Algorithm

Assume the following propositions

1. $F(m,k) \rightarrow \langle c \rangle (F(m,k-1) \wedge F(x))$

1′. $F(m,k) \rightarrow (c,i)(F(m,k-1) \wedge F(x))$

2. $F(m,k) \rightarrow [r*]F(m,k)$

Example: Banach-Knaster Cake Cutting Algorithm

Assume the following propositions

1. $F(m,k) \rightarrow \langle c \rangle (F(m,k-1) \wedge F(x))$

1′. $F(m,k) \rightarrow (c,i)(F(m,k-1) \wedge F(x))$

2. $F(m,k) \rightarrow [r*]F(m,k)$

3. $F(m,k) \rightarrow [c][r*](F(m,k-1) \vee \langle r \rangle (F(m,k-1) \wedge F(x)))$

Example: Banach-Knaster Cake Cutting Algorithm

Assume the following propositions

1. $F(m,k) \rightarrow \langle c \rangle (F(m,k-1) \wedge F(x))$

1'. $F(m,k) \rightarrow (c,i)(F(m,k-1) \wedge F(x))$

2. $F(m,k) \rightarrow [r*]F(m,k)$

3. $F(m,k) \rightarrow [c][r*](F(m,k-1) \vee \langle r \rangle (F(m,k-1) \wedge F(x)))$

4. $F(x) \rightarrow [a_i]F_i$

Example: Banach-Knaster Cake Cutting Algorithm

Assume the following propositions

1. $F(m,k) \rightarrow \langle c \rangle (F(m,k-1) \wedge F(x))$

1′. $F(m,k) \rightarrow (c,i)(F(m,k-1) \wedge F(x))$

2. $F(m,k) \rightarrow [r*]F(m,k)$

3. $F(m,k) \rightarrow [c][r*](F(m,k-1) \vee \langle r \rangle (F(m,k-1) \wedge F(x)))$

4. $F(x) \rightarrow [a_i]F_i$

There are tacit assumptions of relevance, e.g. that r and c can only affect statements in which m or x occurs.

We assume moreover that $F(m,n)$ is true at the beginning.

Example: Banach-Knaster Cake Cutting Algorithm

The (in)formal proof:

1. We show now that each person p_i has a winning strategy so that if, after the kth cycle, (s)he is still in the game then $F(m, n-k)$ and if (s)he is assigned a piece, then F_i is true.

Example: Banach-Knaster Cake Cutting Algorithm

The (in)formal proof:

1. We show now that each person p_i has a winning strategy so that if, after the kth cycle, (s)he is still in the game then $F(m, n-k)$ and if (s)he is assigned a piece, then F_i is true.

2. This is true at start since $k = 0$, $F(m, n)$ holds and no one yet has a piece.

Example: Banach-Knaster Cake Cutting Algorithm

The (in)formal proof:

1. We show now that each person p_i has a winning strategy so that if, after the kth cycle, (s)he is still in the game then $F(m, n-k)$ and if (s)he is assigned a piece, then F_i is true.

2. This is true at start since $k = 0$, $F(m, n)$ holds and no one yet has a piece.

3. We now consider the inductive step from k to $k+1$. We assume by induction hypothesis that $F(m, n-k)$ holds at this stage.

Example: Banach-Knaster Cake Cutting Algorithm

The (in)formal proof:

1. We show now that each person p_i has a winning strategy so that if, after the kth cycle, (s)he is still in the game then $F(m, n-k)$ and if (s)he is assigned a piece, then F_i is true.

2. This is true at start since $k = 0$, $F(m, n)$ holds and no one yet has a piece.

3. We now consider the inductive step from k to $k+1$. We assume by induction hypothesis that $F(m, n-k)$ holds at this stage.

4. If $i = 1$ then since p_1 (or whoever does the cutting) does the cutting, by (1) and (1') she can achieve $F(m, n-k-1) \wedge F(x)$.

Example: **Banach-Knaster Cake Cutting Algorithm**

5. If no one does an r, she gets x and F_1 will hold since x did not change. If someone does do an r, then by (2), $F(m, n-k-1)$ will still hold and this is **OK** since she will then be participating at the next stage.

Example: Banach-Knaster Cake Cutting Algorithm

5. If no one does an r, she gets x and F_1 will hold since x did not change. If someone does do an r, then by (2), $F(m, n-k-1)$ will still hold and this is **OK** since she will then be participating at the next stage.

6. Let us now consider just one of the other people. The last person p_i to do r (if there is someone who does r) could (by (3)) achieve $F(x)$ and therefore when x is assigned to him, F_1 will hold.

Example: Banach-Knaster Cake Cutting Algorithm

5. If no one does an r, she gets x and F_1 will hold since x did not change. If someone does do an r, then by (2), $F(m, n-k-1)$ will still hold and this is **OK** since she will then be participating at the next stage.

6. Let us now consider just one of the other people. The last person p_i to do r (if there is someone who does r) could (by (3)) achieve $F(x)$ and therefore when x is assigned to him, F_1 will hold.

7. All the other cases are quite analogous, and the induction step goes through. By taking $k = n$ we see that every p_i has the ability to achieve F_i.

Thank you.