$\mathcal{L}^1_{\text{UCS}}$: A Monodic Logic for Bounded Model Checking of Unbounded Client-Server Systems

Ramchandra Phawade[1] Tephilla Prince[1] S. Sheerazuddin[2]

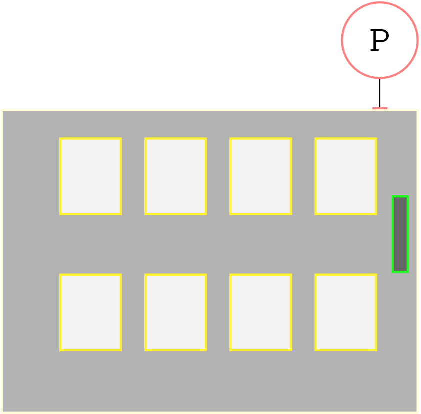
Indian Institute of Technology Dharwad, India

National Institute of Technology Calicut, India

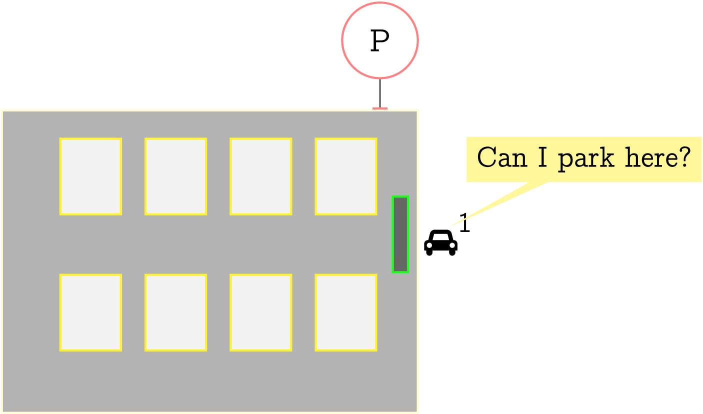
3/3/2023
ICLA'23, IIT Indore

A toy example

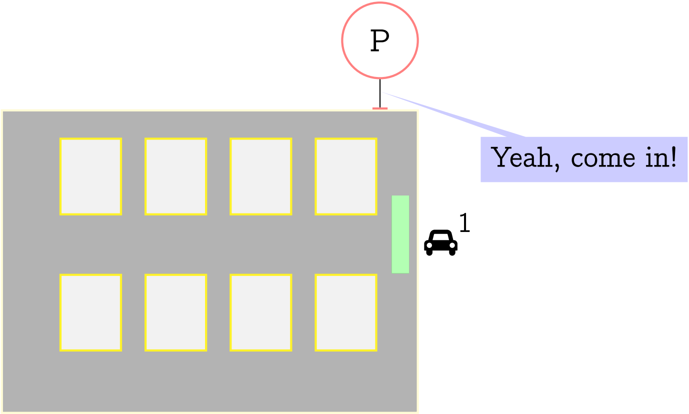
An empty Parking Lot waits for vehicles



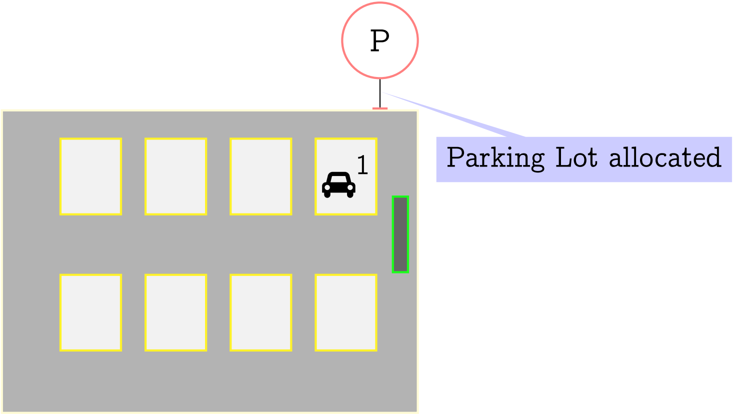
A toy example



A toy example

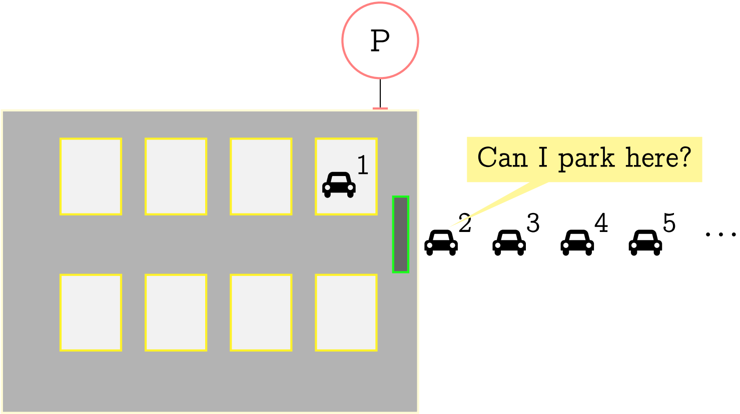


A toy example



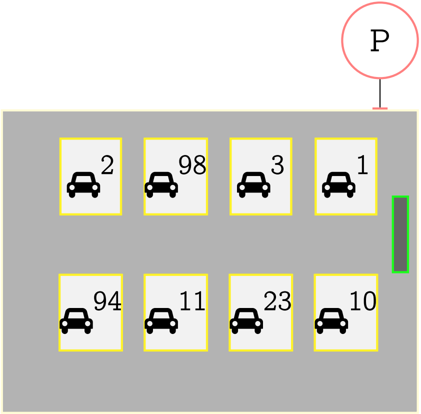
A toy example

Unbounded number of parking requests

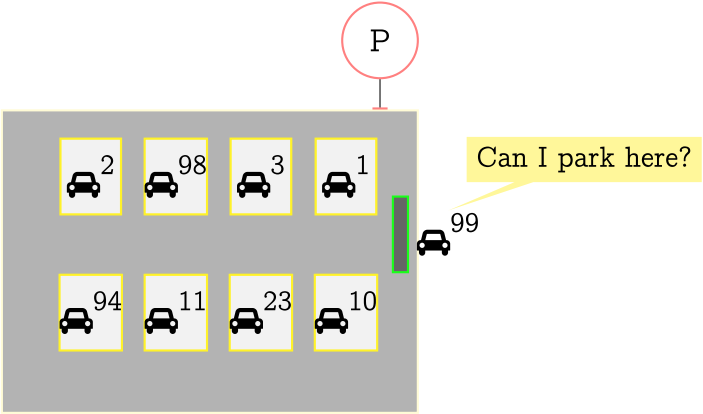


A toy example

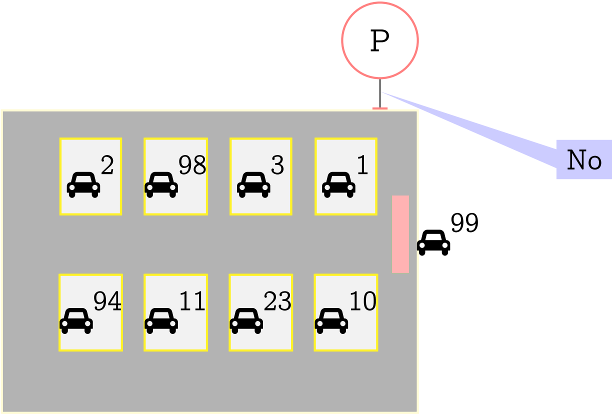
No parking space



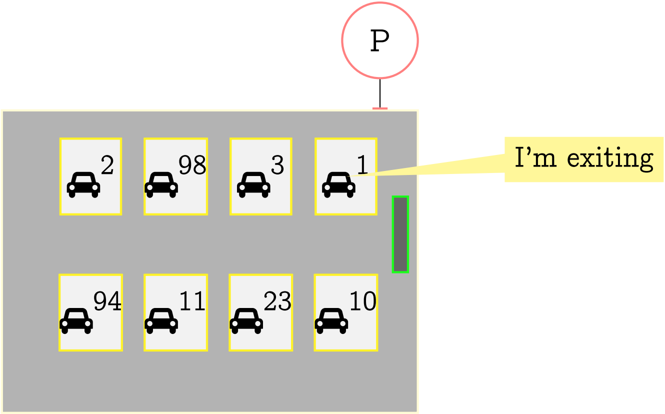
A toy example



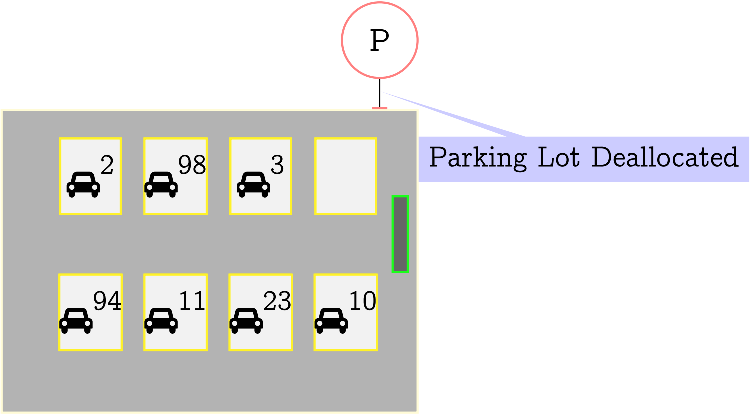
A toy example



A toy example

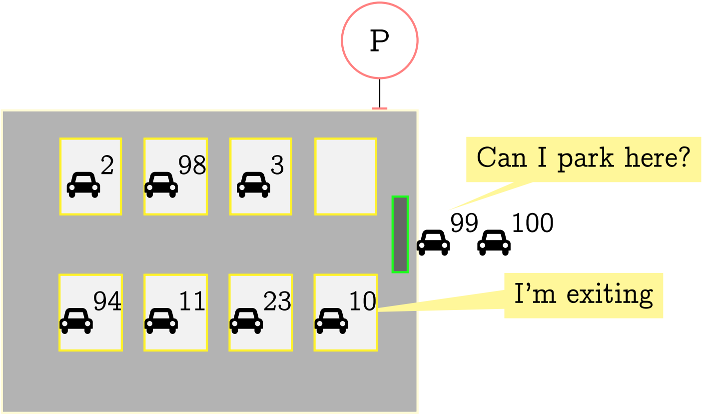


A toy example

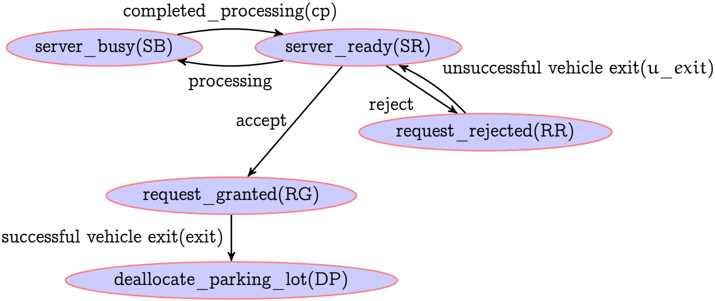


A toy example

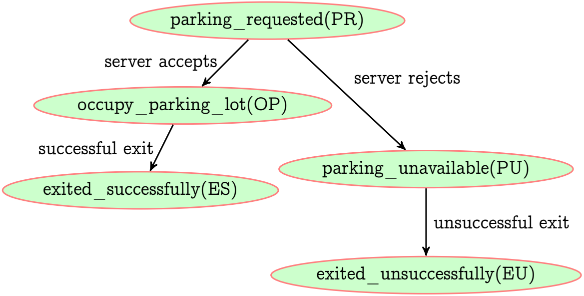
- single server-multiple client system
- unbounded, distinguishable clients of the same type



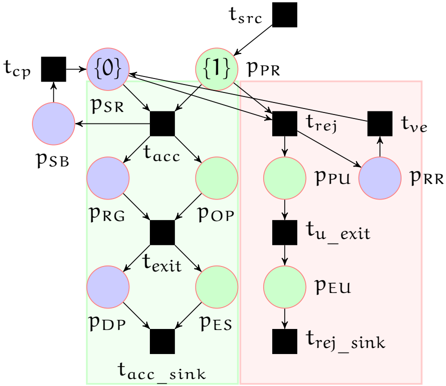
Case Study Parking System: Server behaviour



Case Study Parking System: Client behaviour



Modelling an Unbounded Client-Server (UCS)



Finite sets of atomic client propositions P_c and server propositions P_s:

$$
\begin{aligned}
P_c = \{ & parking\_requested(PR), \\
& occupy\_parking\_lot(OP), \\
& parking\_unavailable(PU), \\
& exit\_successfully(ES), \\
& exited\_unsuccessfully(EU)\} \\
P_s = \{ & server\_ready(SR), \\
& server\_busy(SB), \\
& request\_granted(RG), \\
& request\_rejected(RR), \\
& deallocate\_parking\_lot(DP)\}
\end{aligned}
$$

Question: What properties of the system are we interested in?

Properties

1. When a vehicle requests a parking space, it is always the case that for every vehicle, it eventually exits the system, either successfully after being granted a parking space, or unsuccessfully, when its request is denied.

Properties

1. When a vehicle requests a parking space, it is always the case that for every vehicle, it eventually exits the system, either successfully after being granted a parking space, or unsuccessfully, when its request is denied.

$$G_s(\forall x)\Big(parking\_requested(x) \Rightarrow$$
$$F_c\ (exit\_successfully(x)\ \vee\ exit\_unsuccessfully(x))\Big)$$

Properties

1. When a vehicle requests a parking space, it is always the case that for every vehicle, it eventually exits the system, either successfully after being granted a parking space, or unsuccessfully, when its request is denied.

$$G_s(\forall x)\Big(parking\_requested(x) \Rightarrow \\ F_c\ (exit\_successfully(x)\ \vee\ exit\_unsuccessfully(x))\Big)$$

2. It is always the case that if the client occupies a parking lot, it will eventually exit the parking lot.

Properties

1. When a vehicle requests a parking space, it is always the case that for every vehicle, it eventually exits the system, either successfully after being granted a parking space, or unsuccessfully, when its request is denied.

$$G_s(\forall x)\Big(\mathit{parking\_requested}(x) \Rightarrow$$
$$F_c\ (\mathit{exit\_successfully}(x)\ \vee\ \mathit{exit\_unsuccessfully}(x))\Big)$$

2. It is always the case that if the client occupies a parking lot, it will eventually exit the parking lot.

$$G_s(\forall x)\Big(\mathit{occupy\_parking\_lot}(x) \Rightarrow F_c(\mathit{exit\_successfully}(x))\Big)$$

Properties

3. There may be clients whose requests are rejected.

Properties

3. There may be clients whose requests are rejected.

   $$G_s(\exists x)\big(parking\_requested(x) \wedge F_c(exit\_unsuccessfully(x))\big)$$

Properties

3. There may be clients whose requests are rejected.

   $$G_s(\exists x)\big(parking\_requested(x) \wedge F_c(exit\_unsuccessfully(x))\big)$$

4. There may be clients who have requested for parking and who wait in the parking unavailable state until they are able to exit the system.

Properties

3. There may be clients whose requests are rejected.

$$G_s(\exists x)\big(parking\_requested(x) \wedge F_c(exit\_unsuccessfully(x))\big)$$

4. There may be clients who have requested for parking and who wait in the parking unavailable state until they are able to exit the system.

$$G_s(\exists x)\Big(parking\_requested(x) \wedge$$
$$F_c\big(parking\_unavailable(x)\ U_c\ exit\_unsuccessfully(x)\big)\Big)$$

The proposed logic: The Monodic* Logic $\mathcal{L}^1_{UCS}$

The set of client formulae Δ:

$$\Delta ::= p(x) \mid \neg\alpha \mid \alpha \vee \beta \mid \alpha \wedge \beta \mid X_c\alpha \mid F_c\alpha \mid G_c\alpha \mid \alpha\ U_c\ \beta$$

where $\alpha, \beta \in \Delta$ and $p \in P_c$, the set of atomic client propositions.

The proposed logic: The Monodic* Logic $\mathcal{L}^1_{UCS}$

The set of client formulae Δ:

$$\Delta ::= p(x) \mid \neg\alpha \mid \alpha \vee \beta \mid \alpha \wedge \beta \mid X_c\alpha \mid F_c\alpha \mid G_c\alpha \mid \alpha \; U_c \; \beta$$

where $\alpha, \beta \in \Delta$ and $p \in P_c$, the set of atomic client propositions.

The set of server formulae Ψ:

$$\Psi ::= q \mid \neg\psi \mid (\exists x)\alpha \mid (\forall x)\alpha \mid \psi_1 \vee \psi_2 \mid \psi_1 \wedge \psi_2 \mid X_s\psi \mid F_s\psi \mid G_s\psi \mid \psi_1 \; U_s \; \psi_2$$

where $\psi, \psi_1, \psi_2 \in \Psi$ and $q \in P_s$, the set of atomic server propositions, $\alpha \in \Delta$.

The Monodic Logic $\mathcal{L}^1_{\text{UCS}}$ (contd)

$$\psi = (\exists x)(\exists y)\Big(\big(PR(x) \wedge PR(y)\big) \wedge F_c\big(ES(x) \wedge ES(y)\big)\Big)$$

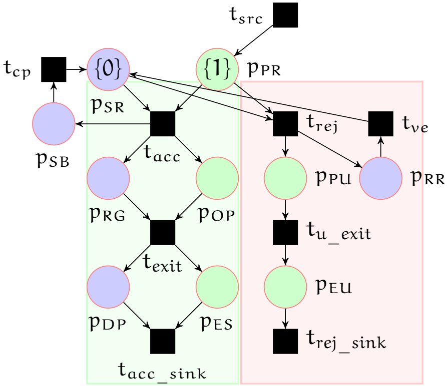
- Is $\psi \in \mathcal{L}^1_{\text{UCS}}$?

The Monodic Logic $\mathcal{L}^1_{\mathsf{UCS}}$ (contd)

$$\psi = (\exists x)(\exists y)\Big(\big(\mathsf{PR}(x) \wedge \mathsf{PR}(y)\big) \wedge \mathbf{F}_c\big(\mathsf{ES}(x) \wedge \mathsf{ES}(y)\big)\Big)$$

- Is $\psi \in \mathcal{L}^1_{\mathsf{UCS}}$?
- Answer: No. **not a well formed formula** in $\mathcal{L}^1_{\mathsf{UCS}}$

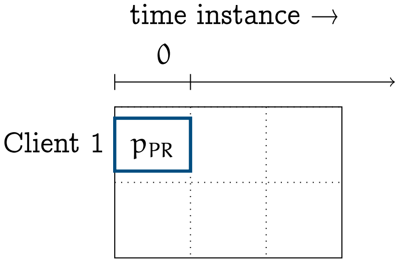
Recall: our model



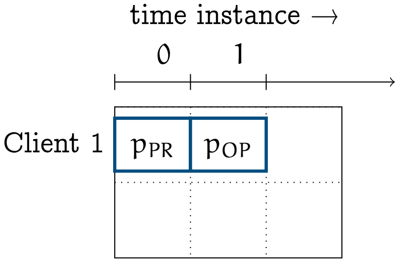
Atomic client propositions P_c and server propositions P_s:

$$
\begin{aligned}
P_c = \{&parking\_requested(PR),\\
&occupy\_parking\_lot(OP),\\
&parking\_unavailable(PU),\\
&exit\_successfully(ES),\\
&exited\_unsuccessfully(EU)\}\\
P_s = \{&server\_ready(SR),\\
&server\_busy(SB),\\
&request\_granted(RG),\\
&request\_rejected(RR),\\
&deallocate\_parking\_lot(DP)\}
\end{aligned}
$$

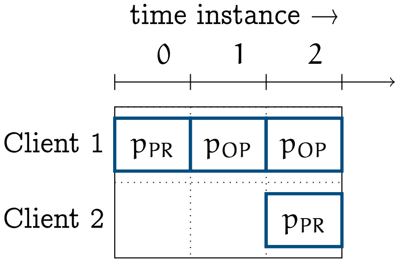
Snapshots of the system + live windows



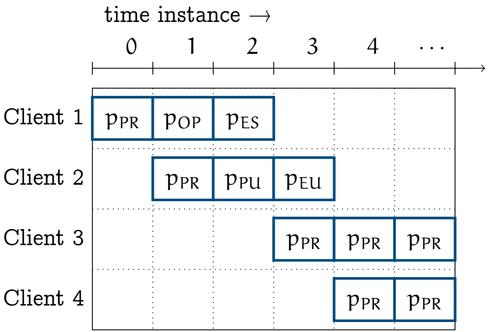
Snapshots of the system + live windows



Snapshots of the system + live windows



Snapshots of the system + live windows (contd)



- 4 distinguishable clients, with overlapping *live windows*, with the bound 5.
- client 1 is at p_{PR} at instance 0. i.e, $PR(x)$ is satisfied in client 1 at instance 0.
- For client 1, the *left boundary* is at instance 0 and its *right boundary* is at instance 2.

Formally speaking...

- Let CN be a countable set of client names.

Formally speaking...

- Let CN be a countable set of client names.
- Let $CS = (Q, \Sigma, \delta, I, F)$ be a finite state machine describing the client behaviour.

Formally speaking...

- Let CN be a countable set of client names.
- Let $CS = (Q, \Sigma, \delta, I, F)$ be a finite state machine describing the client behaviour.
- Let $L : Q \to 2^{P_c}$ be the labelling function of each client state $q \in Q$ in terms of a subset of propositions P_c true in that state.

Formally speaking...

- Let CN be a countable set of client names.
- Let $CS = (Q, \Sigma, \delta, I, F)$ be a finite state machine describing the client behaviour.
- Let $L : Q \to 2^{P_c}$ be the labelling function of each client state $q \in Q$ in terms of a subset of propositions P_c true in that state.
- Let $\mathfrak{Z} : CN \times \mathbb{N}_0 \to Q$ be a partial mapping describing the local state of each client $a \in CN$ at an instance $i \in \mathbb{N}_0$.

Formally speaking...

- Let CN be a countable set of client names.
- Let $CS = (Q, \Sigma, \delta, I, F)$ be a finite state machine describing the client behaviour.
- Let $L : Q \to 2^{P_c}$ be the labelling function of each client state $q \in Q$ in terms of a subset of propositions P_c true in that state.
- Let $\mathfrak{Z} : CN \times \mathbb{N}_0 \to Q$ be a partial mapping describing the local state of each client $a \in CN$ at an instance $i \in \mathbb{N}_0$.

- For instance, $\mathfrak{Z}(a, i) \in q$, means that the local state of each client a at instance i is state q, where $q \in Q$.

Formally speaking...

A model is a triple $M = (\nu, V, \xi)$ where

- ν gives the local behaviour of the server as follows:
  $\nu = \nu_0 \nu_1 \nu_2 \ldots$, where for all $0 \leq i$, $\nu_i \subseteq P_s$

Formally speaking...

A model is a triple $M = (\nu, V, \xi)$ where

- ν gives the local behaviour of the server as follows:
  $\nu = \nu_0 \nu_1 \nu_2 \ldots$, where for all $0 \leq i$, $\nu_i \subseteq P_s$
- V gives the set of live agents (clients) at each instance.
  $V = V_0 V_1 V_2 \ldots$, where for all $0 \leq i$, V_i is a finite subset of CN, gives the set of live agents at the ith instance.
  For every $0 \leq i$, V_i and V_{i+1} satisfy the following properties:
  1. if $V_i \subseteq V_{i+1}$ then for every $a \in V_{i+1} - V_i$ such that $\mathfrak{Z}(a, i+1) \in I$.
  2. if $V_{i+1} \subseteq V_i$ then for every $a \in V_i - V_{i+1}$ such that $\mathfrak{Z}(a, i) \in F$.

Formally speaking...

A model is a triple $M = (\nu, V, \xi)$ where

- ν gives the local behaviour of the server as follows:
  $\nu = \nu_0 \nu_1 \nu_2 \ldots$, where for all $0 \leq i$, $\nu_i \subseteq P_s$
- V gives the set of live agents (clients) at each instance.
  $V = V_0 V_1 V_2 \ldots$, where for all $0 \leq i$, V_i is a finite subset of CN, gives the set of live agents at the ith instance.
  For every $0 \leq i$, V_i and V_{i+1} satisfy the following properties:
  1. if $V_i \subseteq V_{i+1}$ then for every $a \in V_{i+1} - V_i$ such that $\mathfrak{Z}(a, i+1) \in I$.
  2. if $V_{i+1} \subseteq V_i$ then for every $a \in V_i - V_{i+1}$ such that $\mathfrak{Z}(a, i) \in F$.
- $\xi = \xi_0 \xi_1 \xi_2 \cdots$, where for all $0 \leq i$, $\xi_i : V_i \to 2^{P_c}$ gives the properties satisfied by each live agent at ith instance.
  Also, $L(\mathfrak{Z}(a, i)) = \xi_i(a)$.

Semantics

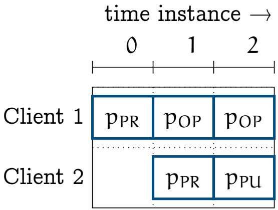
$M, i \models (\exists x)\alpha$ iff $\exists a \in CN,\ a \in V_i$ and $M, [x \mapsto a], i \models_x \alpha$.

Semantics

$M, i \models (\exists x)\alpha$ iff $\exists a \in CN$, $a \in V_i$ and $M, [x \mapsto a], i \models_x \alpha$.
$M, i \models^k (\exists x)\alpha$ iff $\exists a \in CN$, $a \in V_i$ and $M, [x \mapsto a], i \models_x^k \alpha$.

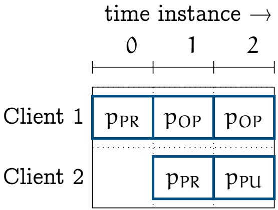
Semantics

$M, i \models (\exists x)\alpha$ iff $\exists a \in CN$, $a \in V_i$ and $M, [x \mapsto a], i \models_x \alpha$.
$M, i \models^k (\exists x)\alpha$ iff $\exists a \in CN$, $a \in V_i$ and $M, [x \mapsto a], i \models_x^k \alpha$.



Semantics

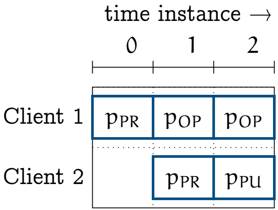
$M, i \models (\exists x)\alpha$ iff $\exists a \in CN$, $a \in V_i$ and $M, [x \mapsto a], i \models_x \alpha$.
$M, i \models^k (\exists x)\alpha$ iff $\exists a \in CN$, $a \in V_i$ and $M, [x \mapsto a], i \models_x^k \alpha$.



Given $\alpha = OP(x) \vee PR(x)$ and $CN = \{1, 2\}$.

Semantics

$M, i \models (\exists x)\alpha$ iff $\exists a \in CN$, $a \in V_i$ and $M, [x \mapsto a], i \models_x \alpha$.
$M, i \models^k (\exists x)\alpha$ iff $\exists a \in CN$, $a \in V_i$ and $M, [x \mapsto a], i \models^k_x \alpha$.

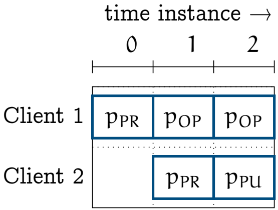


Given $\alpha = OP(x) \vee PR(x)$ and $CN = \{1, 2\}$.

- $M, 0 \models^k (\exists x)(OP(x) \vee PR(x))$.
- $M, 1 \models^k (\exists x)(OP(x) \vee PR(x))$.
- $M, 2 \models^k (\exists x)(OP(x) \vee PR(x))$.

Bounded Semantics

$M, i \models^k (\forall x)\alpha$ iff $\forall a \in CN$, if $a \in V_i$ then $M, [x \mapsto a], i \models^k_x \alpha$



Given $\alpha = ES(x)$ and $CN = \{1, 2\}$.

- $M, 0 \not\models^k (\exists x)(ES(x))$.
- $M, 1 \not\models^k (\exists x)(ES(x))$.
- $M, 2 \not\models^k (\exists x)(ES(x))$.

Bounded Semantics

$M, i \models^k F_s\psi$ iff $\exists j : i \leq j \leq k$, $M, j \models^k \psi$.
Given $\psi = (\exists x)ES(x)$.

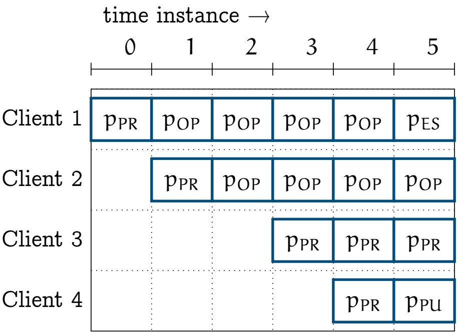


Figure: Snapshot with 4 clients

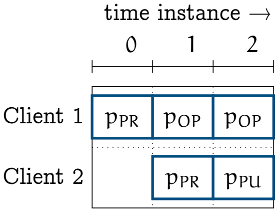


Figure: Snapshot with 2 clients

Bounded Semantics

$M, i \models^k F_s\psi$ iff $\exists j : i \leq j \leq k$, $M, j \models^k \psi$.
Given $\psi = (\exists x)ES(x)$.

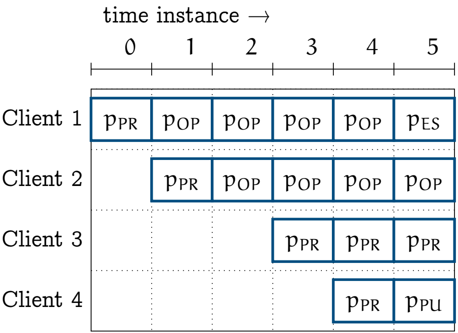


Figure: Snapshot with 4 clients

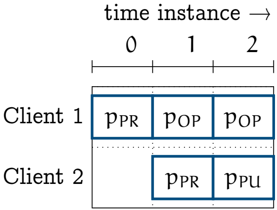


Figure: Snapshot with 2 clients

As shown above $M, 5 \models^k F_s(\exists x)ES(x)$.

Takeaway

- Proposed the logic $\mathcal{L}^1_{\mathrm{UCS}}$ for client-server specifications.
- Modeled the running example using nets.
- Detailed Semantics and encoding are in our paper!
- Currently working on building a model checker to verify $\mathcal{L}^1_{\mathrm{UCS}}$ properties.

Thank you for your attention

Appendix

Bounded Semantics

$M, [x \mapsto a], i \models_x^k F_c \alpha$ iff
$\exists j : i \leq j \leq k,\ a \in V_j$ and $M, [x \mapsto a], j \models_x^k \alpha$.

time instance →	0	1	2	3	4	5
Client 1	p_{PR}	p_{OP}	p_{OP}	p_{OP}	p_{OP}	p_{ES}
Client 2		p_{PR}	p_{OP}	p_{OP}	p_{OP}	p_{OP}
Client 3				p_{PR}	p_{PR}	p_{PR}
Client 4					p_{PR}	p_{PU}

Figure: Snapshot with 4 clients

time instance →	0	1	2
Client 1	p_{PR}	p_{OP}	p_{OP}
Client 2		p_{PR}	p_{PU}

Figure: Snapshot with 2 clients

Given $\alpha = PR(x)$. The above formula is satisfiable for all clients in both figures.