



Validación Experimental de Abstracciones Modales para Contratos Inteligentes

Defensa de Tesis de Licenciatura en Ciencias de la Computación

Autores: Matías Nicolás Incem y Alejandra Alicia Rodríguez

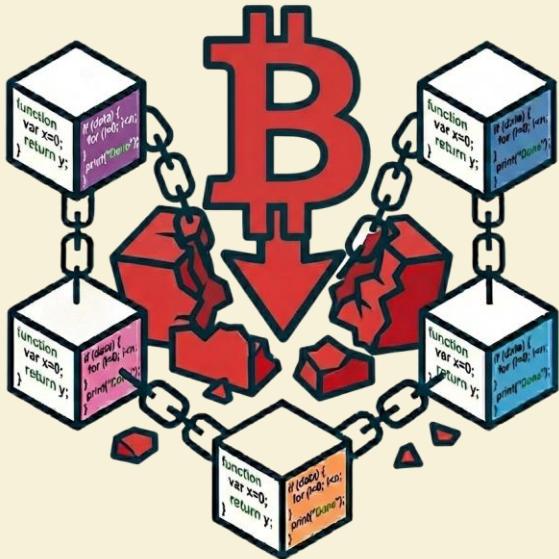
Director: Dr. Diego Garberovetsky

Año: 2025

Motivación

Smart Contracts: El Dilema del Código Inmutable

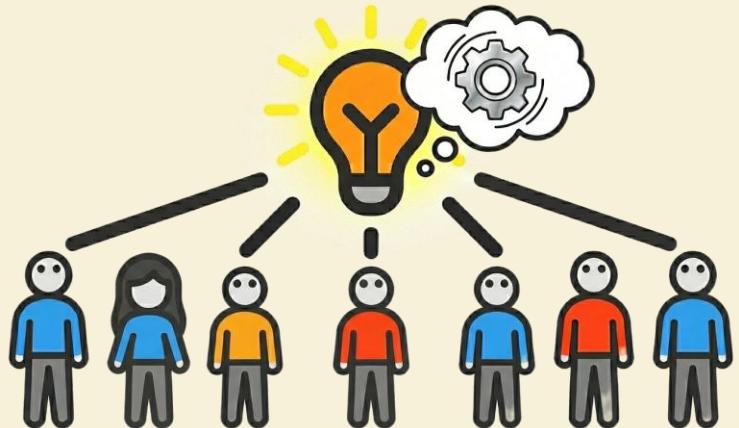
"Un error en el código puede costar millones, y no se puede corregir una vez desplegado."



- Son el **pilar de las Finanzas Descentralizadas (DeFi)**.
- Administran activos de gran valor.
- Correr sobre **Blockchain** hace su **código inmutable (confiable)**.
- Pero su **inmutabilidad** es un arma de doble filo.
- Un bug desplegado es eterno y costoso.
- La **verificación previa al despliegue** es vital.
- Pero a menudo no se detectan los **errores sutiles en la lógica de negocio**.
- Las **herramientas actuales se enfocan en vulnerabilidades clásicas** (overflow, reentrancy), no en la validación profunda del flujo lógico del contrato.

Objetivo

Propuesta del paper base para validar Contratos Inteligentes



Los autores de [Abstracciones Modales para la validación de contratos inteligentes](#) proponen:

- Una nueva herramienta llamada [Alloy4PA](#).
- Dada una especificación formal de un contrato, modela al mismo como un [grafo de estados y transiciones](#). Calificando estas últimas en [posibles o garantizadas](#).
- Ayudando así a [validar el flujo del contrato](#), alineado con [la lógica de negocio](#).

Aporte

¿Qué Abarca Nuestra Tesis?

1. Comprendión



Analizar lo planteado en el paper y entender el contexto desde Blockchain y Predicate Abstractions hasta Solidity y Alloy.

2. Reproducción y Verificación



Replicar y documentar sistemáticamente los experimentos del paper original, en diferentes entornos, usando Alloy4PA para verificar la consistencia de sus hallazgos.

3. Extensión y Validación



Diseñar e implementar nuevos casos de estudio para evaluar la efectividad, limitaciones y uso de Alloy4PA para detectar errores de lógica de negocio en escenarios no explorados.

4. Análisis y Discusión



Discutir la efectividad, limitaciones y usabilidad de Alloy4PA para detectar errores de lógica de negocio en escenarios prácticos.

Fundamentos

Nuestro Ecosistema Técnico

Bases



Blockchain

Una cadena inmutable de bloques que funciona como una base de datos compartida con un histórico irrefutable de información. Su propiedad clave es que la información de un bloque solo puede ser modificada alterando todos los bloques posteriores, lo cual la vuelve prácticamente inmutable.

Lenguaje para



Smart Contracts

Programas que se ejecutan sobre Blockchain y administran activos de gran valor. Permiten establecer acuerdos confiables con menor intervención de intermediarios, pero heredan la inmutabilidad de la plataforma.

Corren sobre

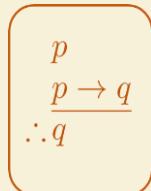
Herramientas y Técnicas

Creación



Solidity

El lenguaje de programación más utilizado para implementar Smart Contracts, diseñado para la red Ethereum y convertido en el estándar del ecosistema.

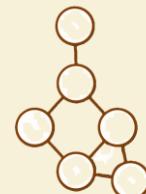


Alloy

Lenguaje y analizador para modelado y verificación. Permite especificar sistemas y verificar automáticamente si dichas especificaciones cumplen propiedades esperadas.

Análisis y Verificación

Usados para análisis y validación



Predicate Abstractions

Técnica para reducir la complejidad de un contrato a un conjunto de propiedades lógicas (predicados). El resultado es un sistema de nodos (estados) y transiciones (funciones) que se puede utilizar para analizar contratos.

Modelado

Abstracciones por Predicados y Abstracciones Modales



Predicate abstraction

Modela un contrato como una serie de estados y transiciones. Representa comportamientos que pueden ocurrir.



Modal abstraction

Evoluciona el modelo anterior para distinguir dos tipos de transiciones: may (puede ocurrir) y must (acción garantizada).



Un ejemplo: ruleta

Mi contrato depende de una ruleta: transiciono del antes al después.

Le pago a cierto jugador según el color del número que sale.

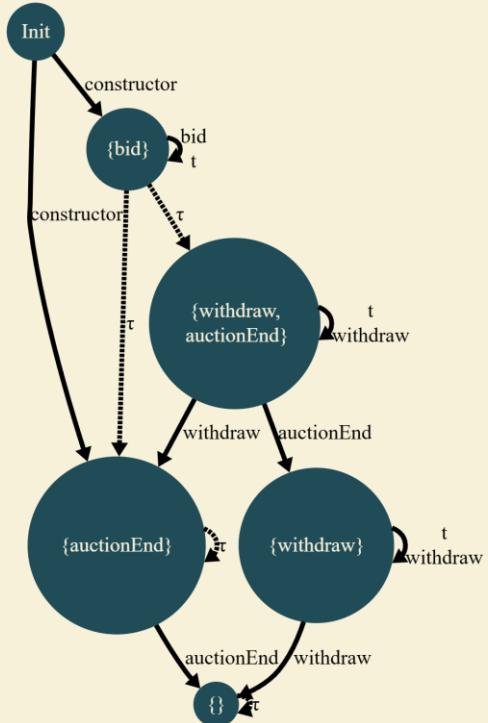
¿Me acordé de resolver el caso del cero, que no es rojo ni negro?

El enfoque modal me va a alertar.

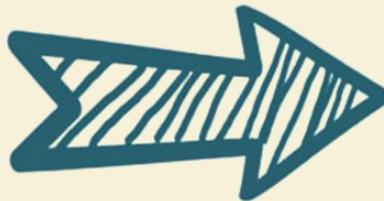


Un ejemplo

Diagrama de transiciones de una subasta

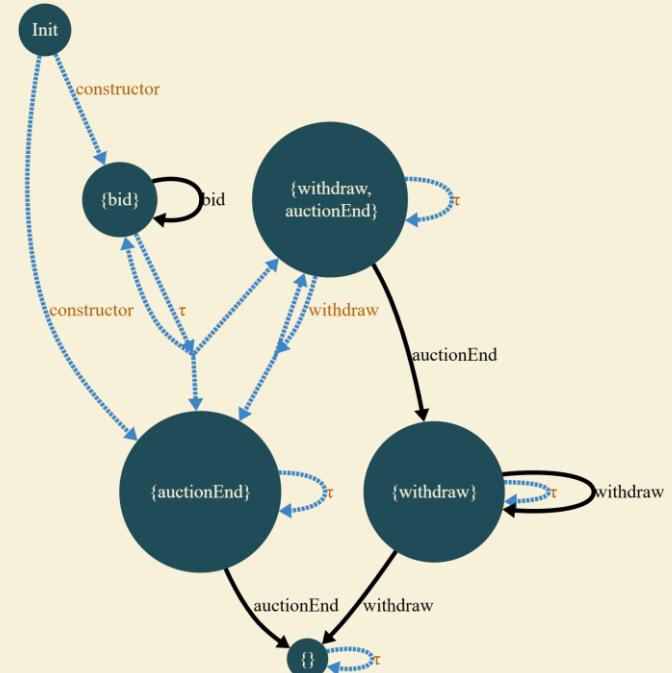


Predicate Abstraction
(todas transiciones *may*)



Evolucionando
el enfoque

Tip: Cuando una transición aparece como *may*, es un posible síntoma de un defecto de lógica, y merece ser revisado.

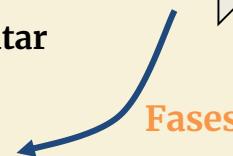
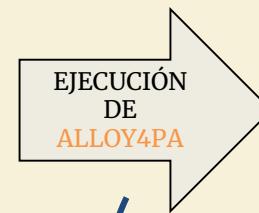
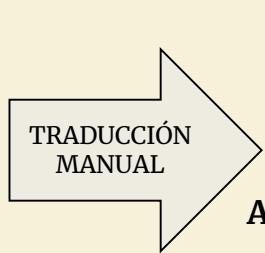


Modal Abstraction
(transiciones *may* y *must*)

Alloy4PA

Presentando la herramienta

Alloy4PA es un programa ejecutable en entornos Java y Docker, que permite analizar las abstracciones modales de contratos inteligentes.



Gráfico

Diagrama de transiciones
Escrito en Graphviz DOT



Estadísticas
Reportes y tiempos
Archivo CSV

1. **EPA** (Enumerated PA):
Genera las combinaciones posibles para el modelo.
2. **BEPA** (Bounded Exhaustive PA):
Clasifica transiciones como *may* o *must*.
3. **SBEPA** (Strong BEPA):
A partir de las *must*, Identifica las *hyper-must*.

Output de Alloy4PA

Salida de consola

```
bash
1 > java -jar Alloy4PA/target/Alloy4PA-1.0.0-fat-jar-with-dependencies.jar examples/
   ↳ Benchmarks\B1\alloy_models\AssetTransfer.als
2 ===== Subjects to generate abstraction:
   ↳=====
3 C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B1\alloy_models\
   ↳ AssetTransfer.als
4 ===== STARTING WITH FILE: C:\Users\USUARIO\
   ↳ Documents\Tesis\Alloy4PA\examples\Benchmarks\B1\alloy_models\AssetTransfer.als
5 =====
6 Configuration loaded successfully from config.properties
7 Max Thread Number: 4
8 Query Timeout Limit (secs): 600
9 Verbose: true
10 Avoid Must Constructor: false
11 Avoid Must Tau: false
12 Avoid Must All: false
13 Allow Address 0x0: true
14 Output Statistics Path: results/
15 Overwrite Subjects: false
16 =====
17 Loading config C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\Benchmarks\B1\
   ↳ alloy_models\AssetTransferConfig.ini
18 Total time generating alloy file with partitions and preds: 0,00 min
19 ===== Parsing+Typechecking C:\Users\USUARIO\Documents\Tesis\Alloy4PA\examples\
   ↳ Benchmarks\B1\AssetTransfer\AssetTransfer_part3.als
20 =====
21 ... Subject,EPA (secs), BEPA (secs), SBEP
22 AssetTransfer,7.988,150.954,0.125
23 Total time all subjects (seconds): 159
24 End.
25 >
```

Archivos generados

Copia de la abstracción



Nombre
AssetTransfer.als
AssetTransfer_bluetransitions_parsed.dot
AssetTransfer_merge_epa_blue.dot
AssetTransfer_merge_epa_superblue.dot
AssetTransfer_original.dot
AssetTransfer_original_reachable.dot
AssetTransfer_originalblue.dot
AssetTransfer_originalblue_reachable.dot
AssetTransfer_originalsuperblue.dot
AssetTransfer_originalsuperblue_reachable.dot
AssetTransfer_part1.als
AssetTransfer_part2.als
AssetTransfer_part3.als
AssetTransfer_queriesTime_BLUE.csv
AssetTransfer_queriesTime_BLUE_raw.csv
AssetTransfer_queriesTime_CLASSIC.csv
AssetTransfer_queriesTime_CLASSIC_raw.csv
AssetTransfer_queriesTime_SUPERBLUE.csv
AssetTransfer_queriesTime_SUPERBLUE_raw.csv
AssetTransfer_superbluetransitions_parsed.dot
AssetTransfer_forked_arrows.dot

Diagramas intermedios



Archivos intermedios



Estadísticas de tiempos



Diagrama de transiciones



Output de Alloy4PA

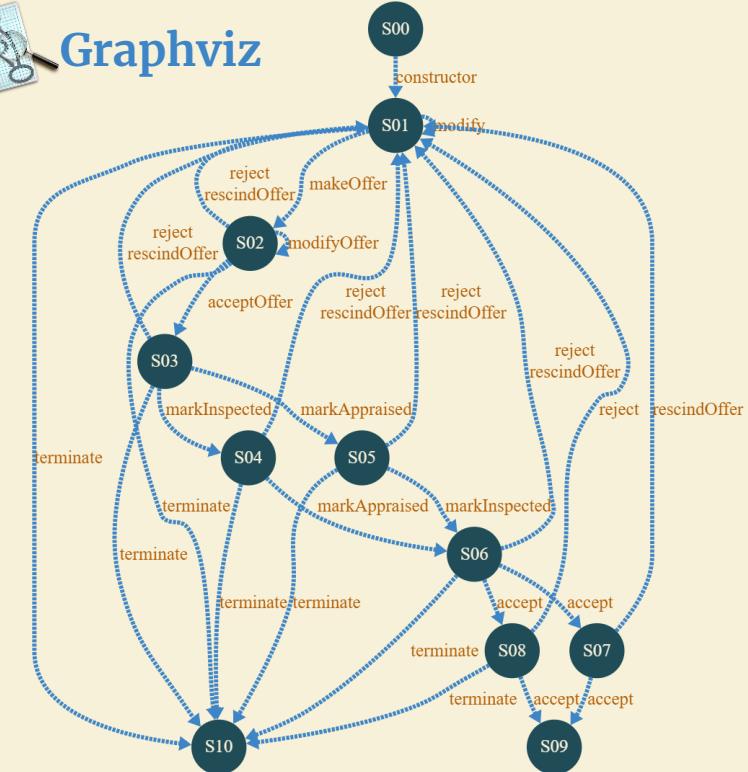
Diagrama de transiciones

Graphviz DOT

```
1 digraph {
2   S00->S01 [label="constructor", style="dotted", color="blue"]
3   S02->S10 [label="terminate", style="dotted", color="blue"]
4   S02->S01 [label="reject\nrescindOffer", style="dotted", color="blue"]
5   S02->S02 [label="modifyOffer", style="dotted", color="blue"]
6   S02->S03 [label="acceptOffer", style="dotted", color="blue"]
7   S01->S02 [label="makeOffer", style="dotted", color="blue"]
8   S01->S01 [label="modify", style="dotted", color="blue"]
9   S01->S10 [label="terminate", style="dotted", color="blue"]
10  S04->S10 [label="terminate", style="dotted", color="blue"]
11  S04->S01 [label="reject\nrescindOffer", style="dotted", color="blue"]
12  S04->S06 [label="markAppraised", style="dotted", color="blue"]
13  S03->S10 [label="terminate", style="dotted", color="blue"]
14  S03->S04 [label="markInspected", style="dotted", color="blue"]
15  S03->S01 [label="reject\nrescindOffer", style="dotted", color="blue"]
16  S03->S05 [label="markAppraised", style="dotted", color="blue"]
17  S06->S08 [label="accept", style="dotted", color="blue"]
18  S06->S07 [label="accept", style="dotted", color="blue"]
19  S06->S10 [label="terminate", style="dotted", color="blue"]
20  S06->S01 [label="reject\nrescindOffer", style="dotted", color="blue"]
21  S05->S10 [label="terminate", style="dotted", color="blue"]
22  S05->S06 [label="markInspected", style="dotted", color="blue"]
23  S05->S01 [label="reject\nrescindOffer", style="dotted", color="blue"]
24  S08->S09 [label="accept", style="dotted", color="blue"]
25  S08->S10 [label="terminate", style="dotted", color="blue"]
26  S08->S01 [label="reject", style="dotted", color="blue"]
27  S07->S09 [label="accept", style="dotted", color="blue"]
28  S07->S01 [label="rescindOffer", style="dotted", color="blue"]
29 }
```



Graphviz



Reproducción

Metodología Seguida

Varios Sistemas Operativos



Instalación de Prerrequisitos



Ejecución de la Herramienta



Solución de Conflictos



Revisión Outputs Benchmarks



RQ1: Chequeo presencia MUST

Benchmark	Metric A	Metric B	Metric C
B1	0.98	1.00	N/A
B2	0.95	0.99	0.97

RQ2: Chequeo Restrición Roles



RQ3: Chequeo Ayuda al Auditar



Reproducción

4 Modalidades de ejecución probadas

- Archivo JAR precompilado:

```
java -jar Alloy4PA/target/Alloy4PA-1.0.0-fat-jar-with-dependencies.jar  
examples/Benchmarks/B1/alloy_models/AssetTransfer.als
```

- Compilando archivo JAR desde fuentes:

```
mvn install:install-file -Dfile="libs/org.alloytools.alloy6.dist.jar" -  
DgroupId="org.alloytools.alloy6.dist" -DartifactId="alloy" -Dversion="6.0.0"  
-Dpackaging="jar"
```

-
- Imagen de Docker provista:

```
docker load -i alloy4pa_amd64.tar  
docker run -it alloy4pa-amd64
```

- Generando imagen de Docker propia:

```
docker build -t alloy4pa-replication .
```



Reproducción

Verificación de los hallazgos del paper



Ejecutamos los benchmarks B1 y B2 del trabajo original.



Nuestros hallazgos (tiempos, número y tipo de transiciones) son consistentes con los publicados.



Confirmación: Las transiciones ‘must’ están presentes en todos los contratos analizados y son más que las ‘may’.

Contrato	Estados	#Trx	#Must	#May	#HM
AssetTransfer	10	32	32	0	0
DigitalLocker	6	12	12	0	0
Auction+EPA	4	15	4	3	4
EscrowVault	4	17	16	1	0
ValidatorAuction+EPA	6	22	7	2	5

Conclusión clave: El enfoque es reproducible y los hallazgos originales son válidos.

Nuevos ejemplos

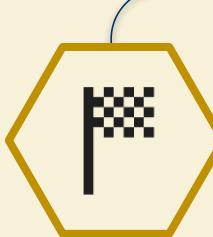
Probando la herramienta en otros contextos

Después de reproducir los resultados originales, buscamos validar la robustez de Alloy4PA en casos no incluidos en el paper.



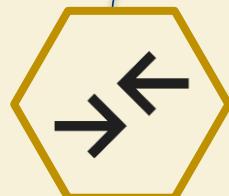
Introduciendo un bug

A partir de un contrato de uno de los benchmarks originales, pensamos en algún cambio mínimo que podamos hacer al código, imitando un descuido de un programador que comete un error.



Objetivo: Alloy4PA expone el bug

La herramienta es capaz de evidenciar bugs en el diagrama de transiciones. Hacemos una predicción del resultado que la herramienta nos debería dar ante el escenario de bug.



Ejecución comparativa

Una vez preparados el contrato original y la variante, ejecutamos Alloy4PA con cada uno, y comparamos los diagramas de transición obtenidos para ver si se cumple nuestra predicción.



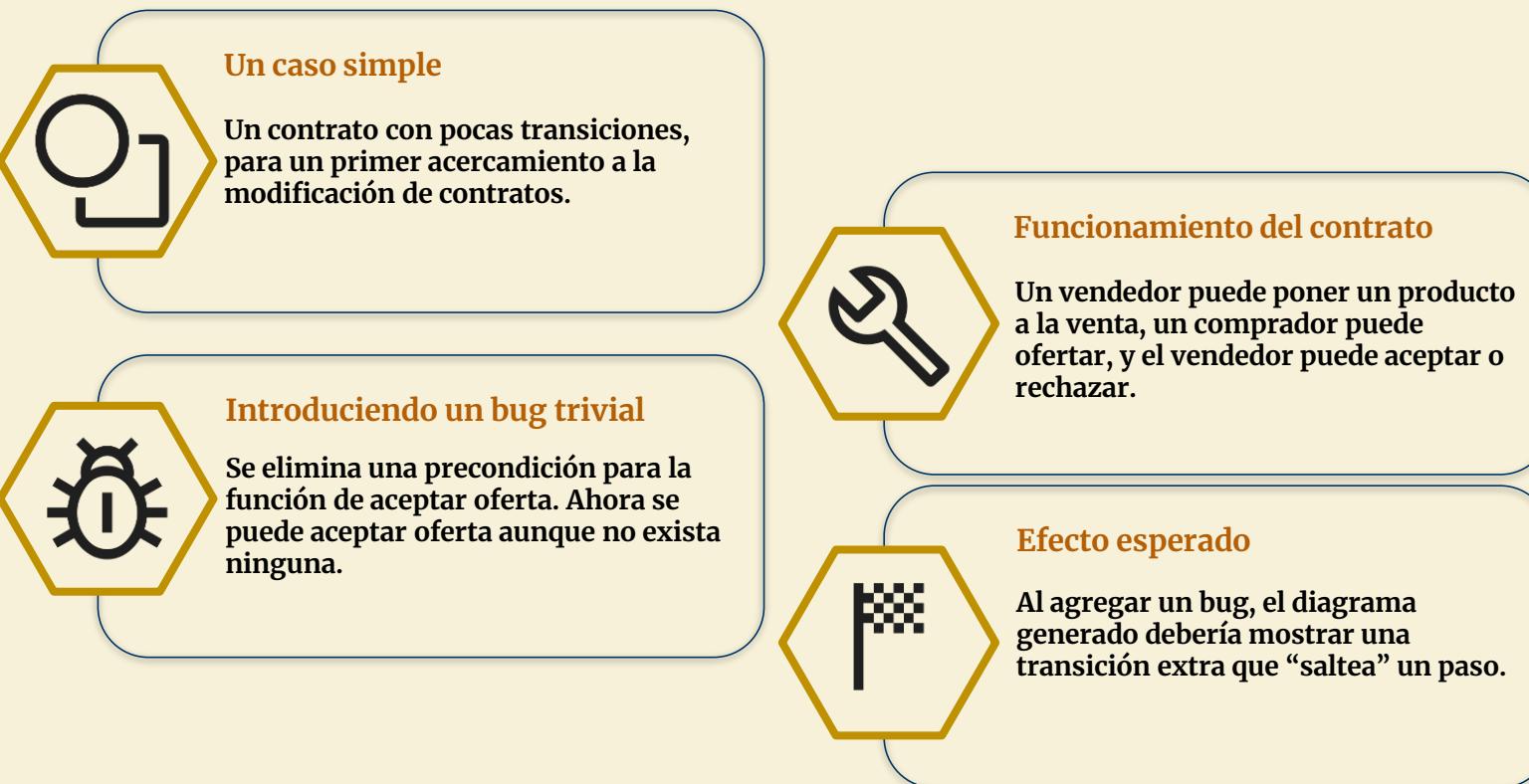
La evidencia esperada

En la comparación, los bugs agregados pueden manifestarse así:

- Aparece una transición incorrecta
- Desaparece una transición esperada
- Una transición *must* se degrada a *may*

Simple Marketplace

Primer acercamiento: Cambio sencillo a contrato sencillo



Simple Marketplace

Resultados: transición anómala

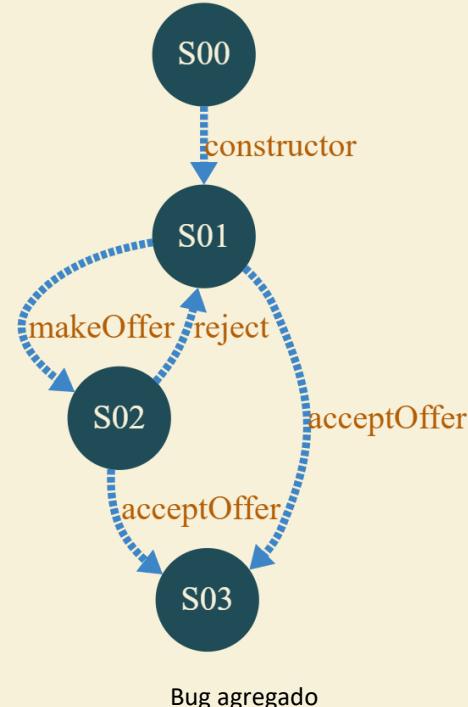
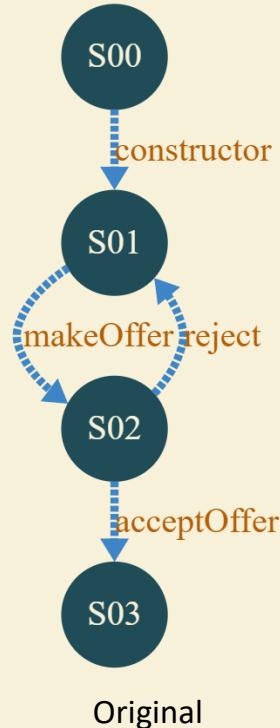
Alloy

```
1 pred pre_acceptOffer[ein: EstadoConcreto] {  
2     some ein._state  
3     // ein._state != ItemAvailable  
4     ein._state != Accepted  
5 }
```

Así modificamos el código para eliminar la precondition.

Al ejecutar, queda verificado que la predicción se cumplió.

En la versión con el bug, aparece la transición que “saltea” un paso del flujo normal.



Rock-Paper-Scissors

Generando un **bug** para que la herramienta lo **revele**



Un caso más ambicioso

Este es un desafío más grande: introducir un bug que convierta una transición *must* en una *may*. Todas las transiciones de este contrato son *must*.



Introduciendo un bug

El contrato incluye un pozo como premio para el ganador, pero no permite llenarlo. Agreguemos una precondition que exija un pozo mayor que 0 para revelar el resultado.



Funcionamiento del contrato

Este contrato modela un juego de piedra, papel, o tijera. Permite que los jugadores 1 y 2 hagan sus jugadas en cualquier orden, y luego revisa el resultado de la partida.



Efecto esperado

Este cambio permite empezar con el pozo en 0, pero solo se puede concluir si es mayor que 0. Algunas ejecuciones no se pueden completar, y por eso la transición de determinar ganador será *may*.

Rock-Paper-Scissors

Primera versión: forzando un pozo no vacío

Solidity

```

1   function determineWinner() public {
2     require(address(this).balance > 0, "Pot must not be empty");
3     //BUG AGREGADO: se exige balance > 0 solo al momento de determinar ganador
4
5     if(p1Choice != -1 && p2Choice != -1) {
6       uint winner = payoffMatrix[uint(p1Choice)][uint(p2Choice)];
7       if(winner == 1) {
8         player1.transfer(address(this).balance);
9       }
10      else if(winner == 2) {
11        player2.transfer(address(this).balance);
12      }
13      else {
14        owner.transfer(address(this).balance);
15      }
16    }
17    else {
18      revert();
19    }
20  }

```

Previo a este punto, se puede avanzar con pozo vacío.

Al momento de revelar el resultado, exigimos un pozo no vacío.

Con este cambio vamos a observar una transición de tipo may.

Alloy

```

1   pred pre_determineWinner[e:EstadoConcreto] {
2     e._balance[Address0x0] > 0
3     //BUG AGREGADO: se exige balance > 0 solo al momento de determinar ganador
4     e._p1Choice != -1 and e._p2Choice != -1
5     e._init = True
6   }

```

Alloy

```

1   pred invariante[e:EstadoConcreto] {
2     e._init = True
3     e._p1Choice >= -1 and e._p1Choice <= 2
4     e._p2Choice >= -1 and e._p2Choice <= 2
5
6     e._payoffMatrix = 0->0->0 + 0->1->2 + 0->2->1 + 1->0->1 + 1->1->0 + 1->2->2 +
7       ↳ 2->0->2 + 2->1->1 + 2->2->0
8
9     some e._balance
10    #e._balance = 4
11
12    // PRECONDICIÓN ELIMINADA: no se exige que todos los balances sean 0
13    // (all a:Address | e._balance[a] = 0)
14    e._player1 != e._player2
15    e._player1 != e._owner
16    e._player2 != e._owner
17    e._player1 != Address0x0
18    e._player2 != Address0x0
19    e._owner != Address0x0
20  }

```

Rock-Paper-Scissors

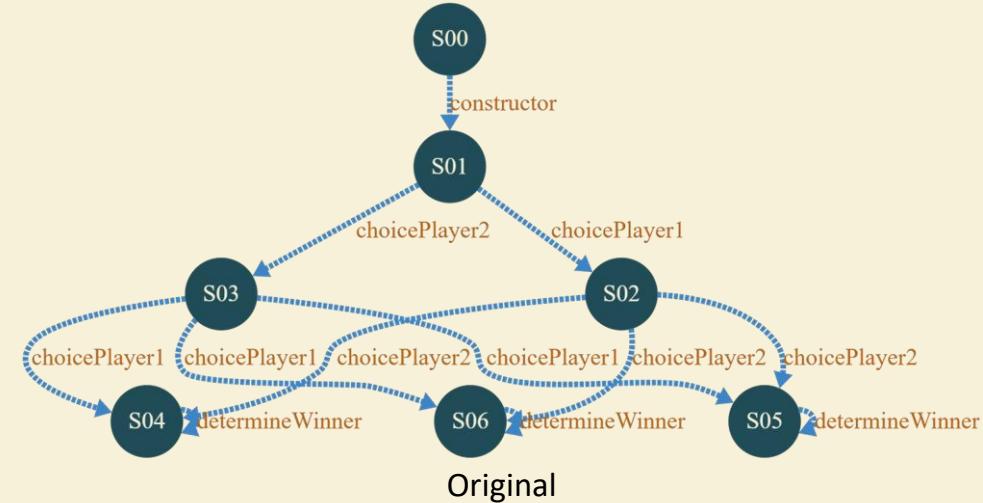
Primer resultado: aparece algo **inesperado**



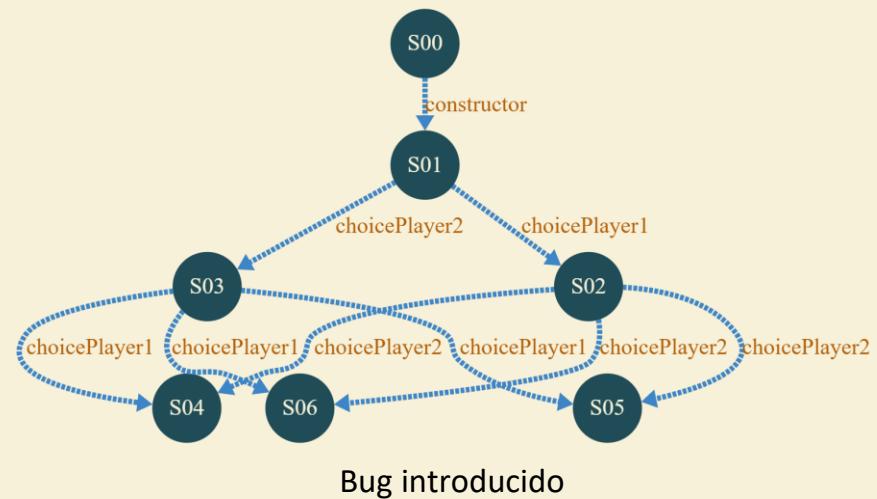
¡Ocurrió algo diferente a nuestra predicción!

La transición de determineWinner no se volvió de tipo *may*.

En vez de eso, desapareció, indicando que es imposible de ejecutar en todo escenario.



Original



Bug introducido

Rock-Paper-Scissors

Segunda versión: corrigiendo la post-condición

Alloy

```

1 pred met_determineWinner[in: EstadoConcreto, out: EstadoConcreto, sender: Address]
2   ↵{
3     (...)

4   //POST
5   let winner = (in._payoffMatrix[in._p1Choice])[in._p2Choice] |
6     (winner = 1) => out._balance = in._balance++Address1->in._balance[Address0x0]
7     else
8       (winner = 2) => out._balance = in._balance++Address2->in._balance[Address0x0]
9       ↵]
10    else
11      (winner = 0) => out._balance = in._balance++in._owner->in._balance[
12        ↵Address0x0]
13
14 out._balance = in._balance++Address0x0->0
15
16 (...)

17 }
```

Post-condición original



La traducción manual a Alloy incluye dos post-condiciones contradictorias.

Exige simultáneamente que el balance del pozo se transfiera al ganador y se mantenga igual, lo cual solo es posible si todos los balances son cero.

Alloy

```

1 // POST
2 let winner = (in._payoffMatrix[in._p1Choice])[in._p2Choice] |
3   let winnerAddress =
4     (winner = 1) => in._player1
5     else (winner = 2) => in._player2
6     else in._owner |
7     // actualizar ganador y vaciar el pozo en una sola igualdad
8     eout._balance =
9       (in._balance) ++ (winnerAddress -> in._balance[Address0x0])
10      ++ (Address0x0 -> 0)
```

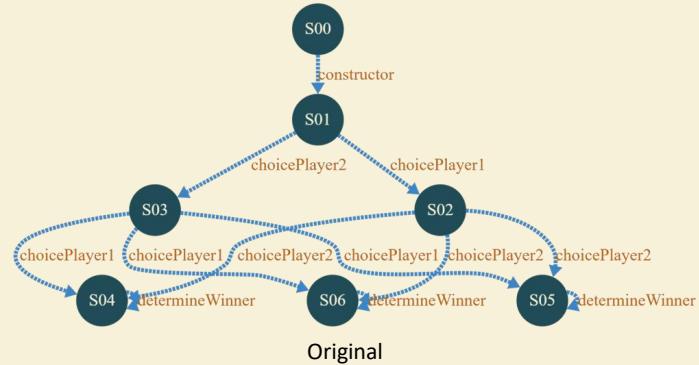
Post-condición arreglada

Este error no impacta al paper original porque ellos simplificaron los balances para ser siempre cero.

Al forzar un balance mayor que cero, la transición se volvió lógicamente imposible.

Rock-Paper-Scissors

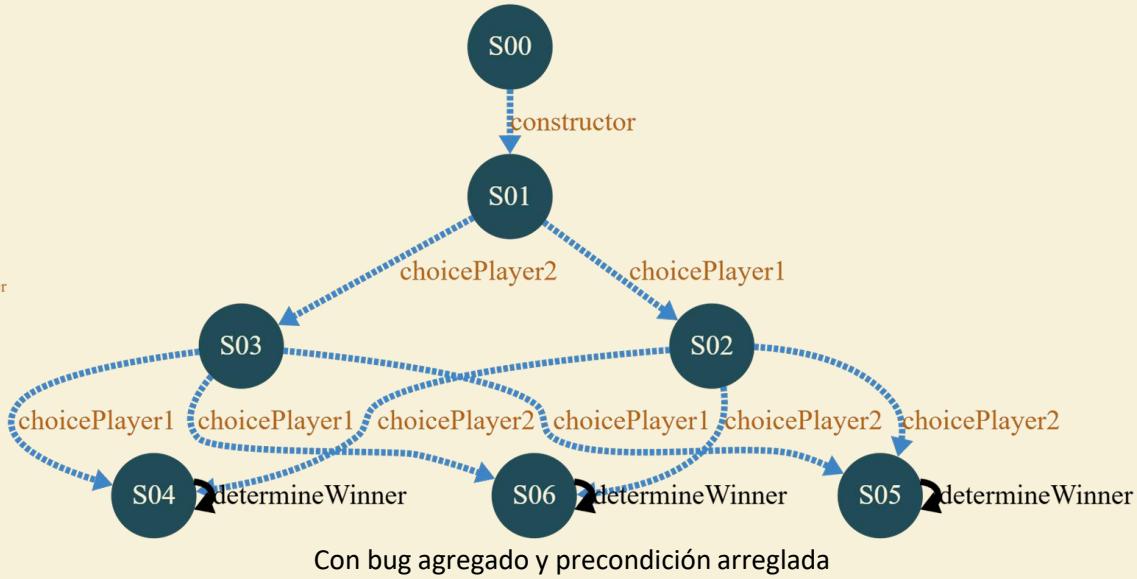
Resultado **final**: la abstracción modal **expone** el bug



Dos limitaciones para determineWinner

Las ejecuciones con pozo 0 no permiten ejecutar la transición.

Las ejecuciones con pozo mayor que 0 permiten ejecutar la transición una única vez, entregando el pozo al ganador.



Ahora sí, la transición reaparece y es de tipo *may*, que es lo que se buscaba originalmente.

AssetTransfer

Contrato de Transferencia de un Activo

Flujo normal de la venta



Vendedor (owner)
Ofrece un activo a un precio solicitado



Comprador (buyer)
Ofrece un precio por el activo



Inspector
Controlador externo verifica transacción



Tasador (appraiser)
Experto externo informa precio



Venta Alcanzada
Vendedor y Comprador Aceptan

Formas de no lograr la transferencia de activo



Rechazar (reject)
El vendedor no acepta la oferta

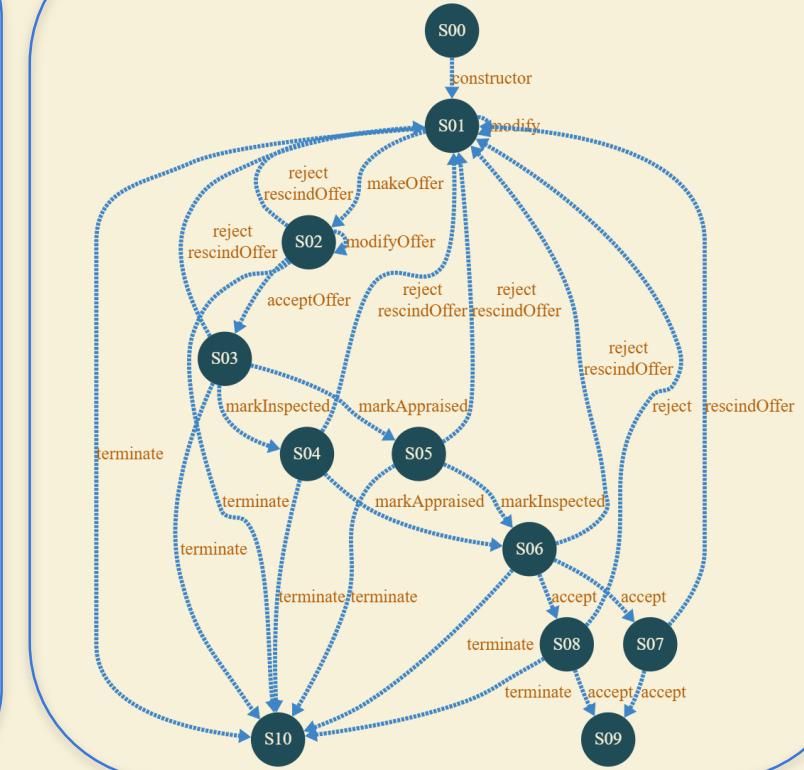


Rescindir (rescind)
El comprador cancela su oferta



Terminar (terminate)
El vendedor decide cancelar la venta

Grafo Alloy4PA



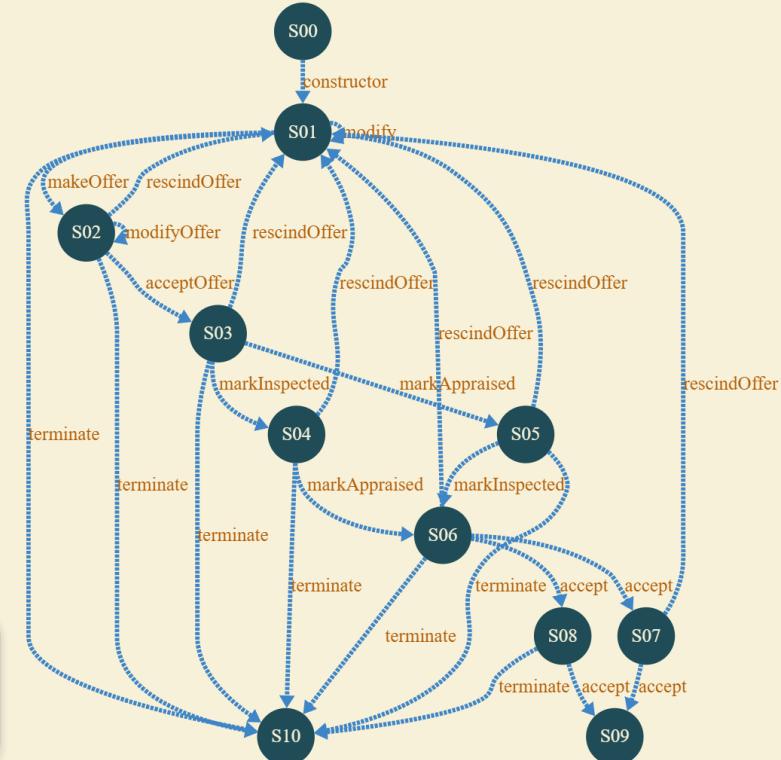
AssetTransfer

Bug 1: Rechazar la oferta no blanquea el comprador

- Añadimos también condición faltante al invariante: Si está en estado activo no hay vendedor.

Introducción de bug
(se muestra especificación en Alloy)

```
pred met_reject[ein: EstadoConcreto, eout: EstadoConcreto, sender: Address] {  
    // Pre  
    ...  
    // Post  
    // eout._instanceBuyer = Address0x0 <-- BUG (comentado)  
    ...  
    // Campos que se mantienen  
    ...  
    eout._instanceBuyer = ein._instanceBuyer <-- BUG  
    ...  
}  
  
pred invariante[e:EstadoConcreto] {  
    ...  
    // Added new condition that was missing  
    e._state = Active implies e._instanceBuyer = Address0x0  
    ...  
}
```



Resultado Clave: Alloy4PA nos ayuda a detectar funciones que se vuelven inalcanzables y desaparecen del modelo.

AssetTransfer

Bug 2: MakeOffer no asigna tasador si el comprador oferta un precio menor o igual al solicitado - EPA=True y EPA=False

Versión Reducida



Menos transiciones

Solo: makeOffer, terminate y accept



Menos estados

Solo: Active, OfferPlaced, Accepted y Terminated



Menos atributos

Sin: descripción ni inspector



Análisis EPA=True

Estados = precondición estado + posibles transiciones siguientes



Análisis EPA=False

Estados = invariante + valor de variable estado

Introducción de bug

(se muestra especificación en Alloy)

```
pred met_makeOffer[in: EstadoConcreto, out:EstadoConcreto, offerPrice:Int] {  
    ...  
    // Post  
    ...  
    // BUG: Solo asigna appraiser si la oferta es MAYOR  
    // que el asking price  
    (offerPrice > in._askingPrice) implies out._instanceAppraiser =  
        appraiser  
    (offerPrice <= in._askingPrice) implies out._instanceAppraiser =  
        in._instanceAppraiser --- BUG  
    ...  
}  
  
pred pre_accept[in: EstadoConcreto] {  
    ...  
    in._instanceAppraiser != Address0x0 // condición faltante  
    ...  
}  
  
pred pre_terminate[in: EstadoConcreto] {  
    ...  
    // Se añade condición faltante  
    in._state = OfferPlaced implies in._instanceAppraiser !=  
        Address0x0  
    ...  
}  
  
pred invariante[e:EstadoConcreto] {  
    ...  
    // Added new condition that was missing  
    e._state = Active implies e._instanceBuyer = Address0x0  
    e._state = Active implies e._instanceAppraiser = Address0x0  
    ...  
}
```

Particiones EPA=True

(se muestra especificación en Alloy)

```
pred pre_<estado>[ein: EstadoConcreto] { //plantilla pre estados  
    ein._init = True  
    ein._state = <Estado>  
}  
pred pre_offerPlaced[in: EstadoConcreto] { //ejemplo pre estado  
    in._init = True  
    in._state = OfferPlaced  
}  
pred pre_offerAccepted[in: EstadoConcreto] { //ejemplo pre estado  
    in._init = True  
    in._state = Accepted  
}  
...  
...
```

Particiones EPA=False

(se muestra especificación en Alloy)

```
pred partitionS00[in: EstadoConcreto] {  
    in._init = True  
}  
pred partitionS0<N>[ein: EstadoConcreto] { //plantilla particiones  
    (invariante[e])  
    ein._state = <Estado>  
}  
pred partitionS02[in: EstadoConcreto] { //ejemplo pre partición  
    (invariante[e])  
    ein._state = OfferPlaced  
}  
...
```

AssetTransfer

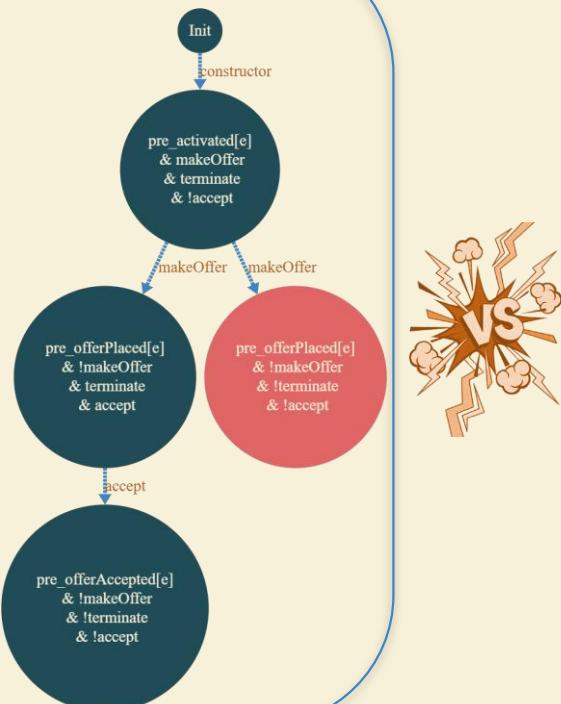
Bug 2: Resultados + Comparación EPA=True vs EPA=False

Resultado EPA=True

Alloy4PA ayuda a detectar flujos que se dividen y conducen a estados sin salida.

EPA=True separa nodos no sólo por estado sino también por las transiciones que son viables ejecutar desde dicho estado. Tiempo de Cómputo mayor (~14 mins). El costo sube exponencialmente con el número de predicados que se le seteen.

El sistema nos entrega un diagnóstico más certero: no se puede salir de ese estado.

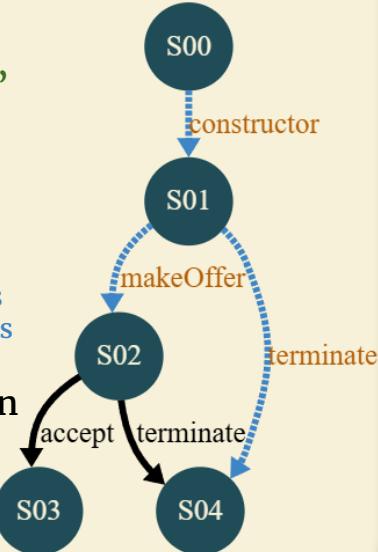


Resultado EPA=False

Alloy4PA logra detectar pérdida de garantías, mediante transiciones degradadas desde “must” a “may”.

EPA=False une en S02 los dos subgrupos de estados, los que tienen el tasador informado con aquellos que no. Tiempo de Cómputo menor (~6½ mins). Es más rápido ya que evalúa sólo las particiones que le pedimos.

El sistema nos alerta de un síntoma: Sabemos que algo impide el progreso, pero no se infiere que es un estado sin salida. Con todas may no se notaría.



Para auditorías robustas ayuda comenzar con EPA=False para identificar subáreas problemáticas rápidamente y EPA=True para investigar esas áreas a fondo y confirmar la naturaleza del error.

Resultados Nuevos Ejemplos



Detección Efectiva

Los nuevos casos de estudio demostraron que Alloy4PA detecta errores de lógica de negocio, visualizándolos como transiciones no siempre posibles ('MAY'), faltantes o bifurcaciones a estados muertos.



Flexibilidad

El enfoque es lo suficientemente sensible para capturar bugs sutiles en diferentes tipos de contratos y lógicas de negocio.

Discusiones

Problemas, limitaciones, oportunidades



Errores encontrados

La herramienta tuvo problemas para leer el formato de rutas de archivo de Windows, y no ejecutó correctamente.

Una ejecución fallida requiere limpiar a mano el directorio de salida para poder ejecutar de nuevo si no se quiere modificar la configuración general.



Experiencia de usuario

La herramienta solo es ejecutable por línea de comandos, lo cual requiere un alto nivel de conocimientos.

Los archivos de salida están mezclados con los intermedios, sería más cómodo recibirlos por separado.

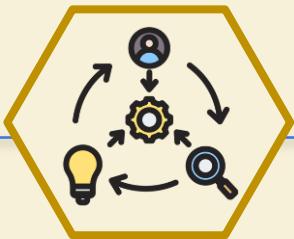


Optimizaciones

Como señalan los autores, la ejecución está enfocada en correctitud, no en tiempo.

Un análisis completo de un solo contrato puede durar media hora.

Hay una oportunidad para mejorar los tiempos de ejecución en futuras versiones.



Esta tesis validó y extendió el enfoque de **Abstracciones Modales**.

Se demostró que es una herramienta reproducible y eficaz para que los auditores identifiquen comportamientos no garantizados y fallas de lógica en contratos inteligentes.

La distinción `may`/`must` es fundamental para este análisis. Especialmente frente a los tiempos requeridos para análisis EPA.

Conclusiones

Síntesis de la Tesis

Respuestas a las RQs Confirmadas



RQ1
Las transiciones 'must' son prevalentes.



RQ2
Las 'constraint transitions' son útiles para analizar condiciones específicas, análogas a roles. Aunque no armamos casos nuevos con bugs de roles.



RQ3
La distinción may/must es una herramienta clave para la auditoría.

Possible proyección Futura



Automatización de modelos
Desarrollar herramientas para generar automáticamente modelos Alloy desde el código fuente de Solidity. Reduciendo los posibles errores humanos.



Interfaz de Usuario
Desarrollar una GUI para simplificar la configuración y ejecución de los análisis.



Mejora de la Documentación
Elaborar guías completas para reducir la curva de aprendizaje e interpretar los resultados.

Cierre

¿Preguntas?

Gracias



Para facilitar la trazabilidad, replicación y verificación independiente, todos los artefactos generados durante esta tesis se encuentran disponibles públicamente en:
github.com/alu-rodriguez/tesis-incem-rodriguez-2025