Verificación de smart contracts en Marlowe para la blockchain Cardano

Julián Ferres

Facultad de Ingeniería
Universidad de Buenos Aires.

5 de agosto de 2022

.UBAfiuba
FACULTAD DE INGENIERÍA

Índice de Contenidos

1 Introducción
- Blockchains, Criptomonedas y Smart contracts
- Cardano
- ACTUS
- Verificación formal de software

2 Escribiendo contratos ACTUS en Cardano
- Notación del estándar ACTUS
- Contratos en Cardano

3 Verificando propiedades en contratos en Marlowe
- El modelo de Marlowe
- Pruebas sencillas sobre contratos específicos
- Warnings en Auction

4 Posibles temas de desarrollo futuro

.UBAfiuba
FACULTAD DE INGENIERÍA

Índice de contenidos

1 Introducción
- Blockchains, Criptomonedas y Smart contracts
- Cardano
- ACTUS
- Verificación formal de software

2 Escribiendo contratos ACTUS en Cardano
- Notación del estándar ACTUS
- Contratos en Cardano

3 Verificando propiedades en contratos en Marlowe
- El modelo de Marlowe
- Pruebas sencillas sobre contratos específicos
- Warnings en Auction

4 Posibles temas de desarrollo futuro

.UBAfiuba
FACULTAD DE INGENIERÍA

Índice de contenidos

1 Introducción
- Blockchains, Criptomonedas y Smart contracts
- Cardano
- ACTUS
- Verificación formal de software

2 Escribiendo contratos ACTUS en Cardano
- Notación del estándar ACTUS
- Contratos en Cardano

3 Verificando propiedades en contratos en Marlowe
- El modelo de Marlowe
- Pruebas sencillas sobre contratos específicos
- Warnings en Auction

4 Posibles temas de desarrollo futuro

.UBAfiuba FACULTAD DE INGENIERÍA

Cadenas de bloques o *Blockchains*

Las cadenas de bloques (*blockchains*), son **estructuras de datos** donde la **información se divide en conjuntos (bloques)** que cuentan con información adicional relativa a bloques previos de la cadena.

Cadenas de bloques o *Blockchains*

Las cadenas de bloques (*blockchains*), son **estructuras de datos** donde la **información se divide en conjuntos (bloques)** que cuentan con información adicional relativa a bloques previos de la cadena.

En dichos bloques, mediante técnicas criptográficas, la información solo puede ser alterada modificando todos los bloques anteriores.

Cadenas de bloques o *Blockchains*

Las cadenas de bloques (*blockchains*), son **estructuras de datos** donde la **información se divide en conjuntos (bloques)** que cuentan con información adicional relativa a bloques previos de la cadena.

En dichos bloques, mediante técnicas criptográficas, la información solo puede ser alterada modificando todos los bloques anteriores.

Esta propiedad facilita su aplicación en un entorno distribuido, de manera tal que la cadena de bloques puede modelar un **registro histórico irrefutable de información**.

Cadenas de bloques o *Blockchains*

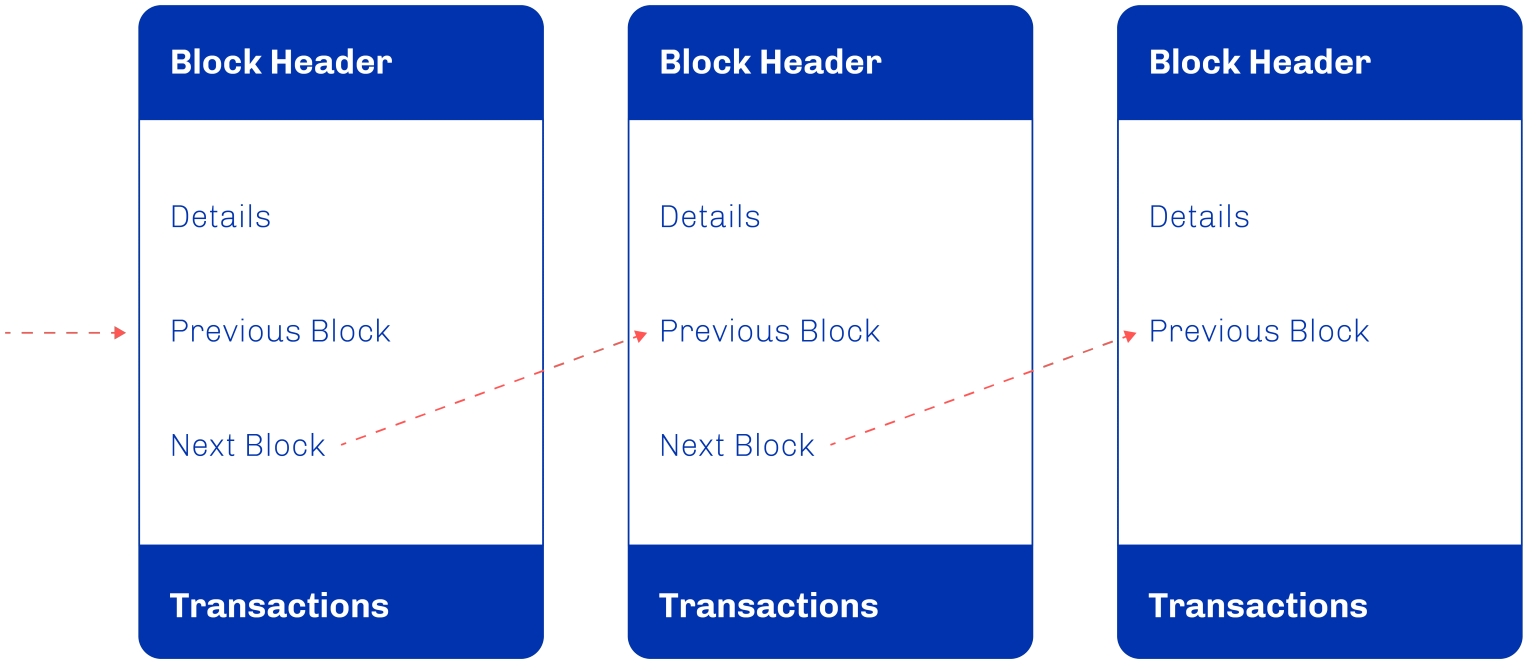
Las cadenas de bloques (*blockchains*), son **estructuras de datos** donde la **información se divide en conjuntos (bloques)** que cuentan con información adicional relativa a bloques previos de la cadena.

En dichos bloques, mediante técnicas criptográficas, la información solo puede ser alterada modificando todos los bloques anteriores.

Esta propiedad facilita su aplicación en un entorno distribuido, de manera tal que la cadena de bloques puede modelar un **registro histórico irrefutable de información**.

Esto ha permitido la **implementación de un registro contable** o ledger distribuido que soporta y garantiza la **seguridad de transacciones y dinero digital**.

.UBAfiuba
FACULTAD DE INGENIERÍA



Representación simplificada de los datos en un bloque de la cadena. Extraída de [Brünjes and Vinogradova, 2019].

Criptomonedas

Las criptomonedas son **activos digitales que se almacenan en el ledger** y están diseñadas para servir como medio de intercambio de bienes o servicios.

Criptomonedas

Las criptomonedas son **activos digitales que se almacenan en el ledger** y están diseñadas para servir como medio de intercambio de bienes o servicios.

Las blockchain son utilizadas como **tecnología subyacente** para la creación de criptomonedas en un entorno descentralizado.

Criptomonedas

Las criptomonedas son **activos digitales que se almacenan en el ledger** y están diseñadas para servir como medio de intercambio de bienes o servicios.

Las blockchain son utilizadas como **tecnología subyacente** para la creación de criptomonedas en un entorno descentralizado.

La creación, seguridad y verificación de las criptomonedas y transacciones son propiedades que debe garantizar la blockchain correspondiente.

Criptomonedas

Las criptomonedas son **activos digitales que se almacenan en el ledger** y están diseñadas para servir como medio de intercambio de bienes o servicios.

Las blockchain son utilizadas como **tecnología subyacente** para la creación de criptomonedas en un entorno descentralizado.

La creación, seguridad y verificación de las criptomonedas y transacciones son propiedades que debe garantizar la blockchain correspondiente.

El precio de la criptomoneda no está controlado por un gobierno o una institución financiera centralizada.

Smart contracts

Un contrato inteligente (*smart contract*) es un **acuerdo digital automatizado**.

Smart contracts

Un contrato inteligente (*smart contract*) es un **acuerdo digital automatizado**.

Están escritos en código y **rastrean, verifican y ejecutan las transacciones de un contrato** entre varias partes.

Smart contracts

Un contrato inteligente (*smart contract*) es un **acuerdo digital automatizado**.

Están escritos en código y **rastrean, verifican y ejecutan las transacciones de un contrato** entre varias partes.

Las transacciones del contrato se ejecutan automáticamente cuando se cumplen las condiciones predeterminadas. Esencialmente, **un contrato inteligente es un programa cuyas entradas y salidas son acciones en una cadena de bloques**.

Smart contracts

Un contrato inteligente (*smart contract*) es un **acuerdo digital automatizado**.

Están escritos en código y **rastrean, verifican y ejecutan las transacciones de un contrato** entre varias partes.

Las transacciones del contrato se ejecutan automáticamente cuando se cumplen las condiciones predeterminadas. Esencialmente, **un contrato inteligente es un programa cuyas entradas y salidas son acciones en una cadena de bloques**.

Los smart contracts no requieren las acciones o la presencia de terceros. El código del contrato inteligente se almacena y distribuye en la *blockchain*, lo que lo hace **transparente e irreversible**.

.UBAfiuba
FACULTAD DE INGENIERÍA

Índice de contenidos

1 Introducción
- Blockchains, Criptomonedas y Smart contracts
- Cardano
- ACTUS
- Verificación formal de software

2 Escribiendo contratos ACTUS en Cardano
- Notación del estándar ACTUS
- Contratos en Cardano

3 Verificando propiedades en contratos en Marlowe
- El modelo de Marlowe
- Pruebas sencillas sobre contratos específicos
- Warnings en Auction

4 Posibles temas de desarrollo futuro

.UBAfiuba FACULTAD DE INGENIERÍA

Cardano es una plataforma blockchain descentralizada de tercera generación, cuya criptomoneda es llamada *ada*.

Cardano es una plataforma blockchain descentralizada de tercera generación, cuya criptomoneda es llamada *ada*.

- **La primera generación** de blockchains (por ejemplo Bitcoin) ofrece ledgers descentralizados para la transferencia segura de criptomonedas.

Cardano es una plataforma blockchain descentralizada de tercera generación, cuya criptomoneda es llamada *ada*.

- **La primera generación** de blockchains (por ejemplo Bitcoin) ofrece ledgers descentralizados para la transferencia segura de criptomonedas.

  Sin embargo, tales *blockchains* no proporcionaron un entorno funcional para la liquidación de acuerdos complejos.

Cardano es una plataforma blockchain descentralizada de tercera generación, cuya criptomoneda es llamada *ada*.

- **La primera generación** de blockchains (por ejemplo Bitcoin) ofrece ledgers descentralizados para la transferencia segura de criptomonedas.

  Sin embargo, tales *blockchains* no proporcionaron un entorno funcional para la liquidación de acuerdos complejos.

- **La segunda generación** (por ejemplo Ethereum) proporcionó soluciones mejoradas para redactar y ejecutar contratos inteligentes, desarrollar aplicaciones y crear diferentes tipos de tokens.

Cardano es una plataforma blockchain descentralizada de tercera generación, cuya criptomoneda es llamada *ada*.

- **La primera generación** de blockchains (por ejemplo Bitcoin) ofrece ledgers descentralizados para la transferencia segura de criptomonedas.

  Sin embargo, tales *blockchains* no proporcionaron un entorno funcional para la liquidación de acuerdos complejos.

- **La segunda generación** (por ejemplo Ethereum) proporcionó soluciones mejoradas para redactar y ejecutar contratos inteligentes, desarrollar aplicaciones y crear diferentes tipos de tokens.

  Sin embargo, la segunda generación de cadenas de bloques a menudo enfrenta problemas en términos de escalabilidad.

Cardano como *blockchain* de 3ra generación

Combina las propiedades de las generaciones anteriores y ofrece las siguientes propiedades:

Cardano como *blockchain* de 3ra generación

Combina las propiedades de las generaciones anteriores y ofrece las siguientes propiedades:

- **Escalabilidad**: Rendimiento de transacciones, escala de datos y ancho de banda de la red.

Cardano como *blockchain* de 3ra generación

Combina las propiedades de las generaciones anteriores y ofrece las siguientes propiedades:

- **Escalabilidad**: Rendimiento de transacciones, escala de datos y ancho de banda de la red.
- **Funcionalidad**: Además del procesamiento de transacciones, la cadena de bloques proporciona todos los medios para la liquidación de acuerdos comerciales.

Cardano como *blockchain* de 3ra generación

Combina las propiedades de las generaciones anteriores y ofrece las siguientes propiedades:

- **Escalabilidad**: Rendimiento de transacciones, escala de datos y ancho de banda de la red.
- **Funcionalidad**: Además del procesamiento de transacciones, la cadena de bloques proporciona todos los medios para la liquidación de acuerdos comerciales.
- **Desarrollo e Integración**: Es sencillo integrar pagos con otras cadenas.

Ada como criptomoneda de Cardano

Ada es la moneda nativa o principal en Cardano. Esto significa que ada es la principal unidad de pago.

Ada como criptomoneda de Cardano

Ada es la moneda nativa o principal en Cardano. Esto significa que ada es la principal unidad de pago.

Sin embargo, también admite la creación de **tokens nativos**: activos digitales que se crean para fines específicos.

Ada como criptomoneda de Cardano

Ada es la moneda nativa o principal en Cardano. Esto significa que ada es la principal unidad de pago.

Sin embargo, también admite la creación de **tokens nativos**: activos digitales que se crean para fines específicos.

Esto permite a los usuarios, desarrolladores y empresas usar a Cardano para crear tokens que representen valor.

Ada como criptomoneda de Cardano

Ada es la moneda nativa o principal en Cardano. Esto significa que ada es la principal unidad de pago.

Sin embargo, también admite la creación de **tokens nativos**: activos digitales que se crean para fines específicos.

Esto permite a los usuarios, desarrolladores y empresas usar a Cardano para crear tokens que representen valor.

Un token puede ser **fungible** (intercambiable) o **no fungible** (único) y actuar como unidad de pago, recompensa, activo comercial o contenedor de información.

.UBAfiuba FACULTAD DE INGENIERÍA

El lenguaje Marlowe

Marlowe es un lenguaje de dominio específico creado por IOHK. El mismo posee **pocas sentencias que describen el comportamiento de un conjunto fijo y finito de roles en un contrato**.

Los contratos se pueden construir reuniendo una pequeña cantidad de estas sentencias que se pueden usar para **describir y modelar contratos financieros**.

El lenguaje Marlowe

Marlowe es un lenguaje de dominio específico creado por IOHK. El mismo posee **pocas sentencias que describen el comportamiento de un conjunto fijo y finito de roles en un contrato**.

Los contratos se pueden construir reuniendo una pequeña cantidad de estas sentencias que se pueden usar para **describir y modelar contratos financieros**.

Algunos ejemplos incluyen:

- Un contrato que puede realizar un pago a un rol o a una clave pública
- Un contrato que puede esperar una acción por parte de uno de los roles:
  - Un depósito de moneda.
  - Una elección entre un conjunto de opciones.

.UBAfiuba FACULTAD DE INGENIERÍA

Índice de contenidos

1 Introducción
- Blockchains, Criptomonedas y Smart contracts
- Cardano
- ACTUS
- Verificación formal de software

2 Escribiendo contratos ACTUS en Cardano
- Notación del estándar ACTUS
- Contratos en Cardano

3 Verificando propiedades en contratos en Marlowe
- El modelo de Marlowe
- Pruebas sencillas sobre contratos específicos
- Warnings en Auction

4 Posibles temas de desarrollo futuro

.UBAfiuba FACULTAD DE INGENIERÍA

Contratos Financieros

Los contratos financieros son **acuerdos legales entre dos (o más) partes** sobre el **futuro intercambio de dinero**.

Dichos acuerdos legales se definen sin ambigüedades por medio de un conjunto de términos y lógica contractual.

Contratos Financieros

Los contratos financieros son **acuerdos legales entre dos (o más) partes** sobre el **futuro intercambio de dinero**.

Dichos acuerdos legales se definen sin ambigüedades por medio de un conjunto de términos y lógica contractual.

Como resultado, los mismos pueden describirse matemáticamente y representarse mediante algoritmos.

En general, **el intercambio de flujos de efectivo** entre partes **sigue ciertos patrones**.

En general, **el intercambio de flujos de efectivo** entre partes **sigue ciertos patrones**.

Un patrón típico es un contrato de préstamo:

En general, **el intercambio de flujos de efectivo** entre partes **sigue ciertos patrones**.

Un patrón típico es un contrato de préstamo:

> *"Se entrega un monto de dinero inicial, a cambio de pagos de intereses cíclicos y la devolución del dinero inicial en el vencimiento del contrato."*

En general, **el intercambio de flujos de efectivo** entre partes **sigue ciertos patrones**.

Un patrón típico es un contrato de préstamo:

> *"Se entrega un monto de dinero inicial, a cambio de pagos de intereses cíclicos y la devolución del dinero inicial en el vencimiento del contrato."*

Si bien los pagos son fijos, **existen muchas variantes** que determinan cómo se programan y/o pagan los pagos de intereses cíclicos:

En general, **el intercambio de flujos de efectivo** entre partes **sigue ciertos patrones**.

Un patrón típico es un contrato de préstamo:

> *"Se entrega un monto de dinero inicial, a cambio de pagos de intereses cíclicos y la devolución del dinero inicial en el vencimiento del contrato."*

Si bien los pagos son fijos, **existen muchas variantes** que determinan cómo se programan y/o pagan los pagos de intereses cíclicos:

- Los pagos de intereses pueden ser mensuales, anuales, mediante períodos arbitrarios.

En general, **el intercambio de flujos de efectivo** entre partes **sigue ciertos patrones**.

Un patrón típico es un contrato de préstamo:

> *"Se entrega un monto de dinero inicial, a cambio de pagos de intereses cíclicos y la devolución del dinero inicial en el vencimiento del contrato."*

Si bien los pagos son fijos, **existen muchas variantes** que determinan cómo se programan y/o pagan los pagos de intereses cíclicos:

- Los pagos de intereses pueden ser mensuales, anuales, mediante períodos arbitrarios.
- Las tasas pueden ser de fijas o variables.

.UBAfiuba FACULTAD DE INGENIERÍA

En general, **el intercambio de flujos de efectivo** entre partes **sigue ciertos patrones**.

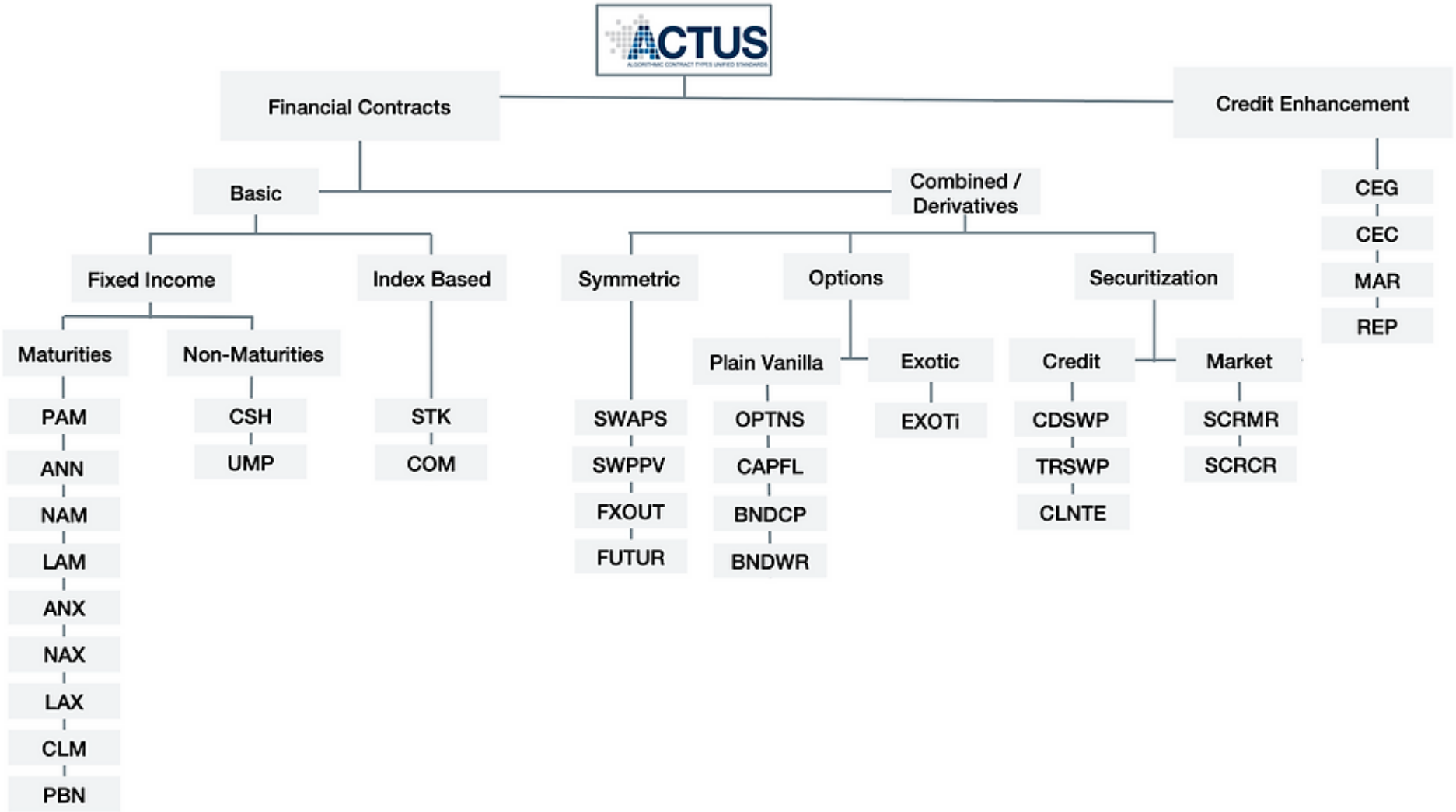
Un patrón típico es un contrato de préstamo:

> *"Se entrega un monto de dinero inicial, a cambio de pagos de intereses cíclicos y la devolución del dinero inicial en el vencimiento del contrato."*

Si bien los pagos son fijos, **existen muchas variantes** que determinan cómo se programan y/o pagan los pagos de intereses cíclicos:

- Los pagos de intereses pueden ser mensuales, anuales, mediante períodos arbitrarios.
- Las tasas pueden ser de fijas o variables.
- Pueden usarse diferentes métodos de cálculo de fracciones anuales o que no haya ningún interés.

.UBAfiuba FACULTAD DE INGENIERÍA



Taxonomía ACTUS

.UBAfiuba
FACULTAD DE INGENIERÍA

Índice de contenidos

1 Introducción
- Blockchains, Criptomonedas y Smart contracts
- Cardano
- ACTUS
- Verificación formal de software

2 Escribiendo contratos ACTUS en Cardano
- Notación del estándar ACTUS
- Contratos en Cardano

3 Verificando propiedades en contratos en Marlowe
- El modelo de Marlowe
- Pruebas sencillas sobre contratos específicos
- Warnings en Auction

4 Posibles temas de desarrollo futuro

.UBAfiuba FACULTAD DE INGENIERÍA

Concepto general, herramientas y metodologías

Los asistentes de pruebas formales son herramientas de software diseñadas para ayudar a sus usuarios a realizar pruebas, especialmente en cálculo lógico.

Concepto general, herramientas y metodologías

Los asistentes de pruebas formales son herramientas de software diseñadas para ayudar a sus usuarios a realizar pruebas, especialmente en cálculo lógico.

A diferencia de los tests convencionales, durante una verificación **no se ejecuta el programa a analizar**, es por eso que este tipo de análisis es denominado **estático**.

Cuando un teorema es escrito y probado, el mismo debe será verdadero para cualquier combinación de las variables de entrada.

Concepto general, herramientas y metodologías

Los asistentes de pruebas formales son herramientas de software diseñadas para ayudar a sus usuarios a realizar pruebas, especialmente en cálculo lógico.

A diferencia de los tests convencionales, durante una verificación **no se ejecuta el programa a analizar**, es por eso que este tipo de análisis es denominado **estático**.

Cuando un teorema es escrito y probado, el mismo debe será verdadero para cualquier combinación de las variables de entrada.

La principal fortaleza de los asistentes es que ayudan a desarrollar **pruebas altamente confiables e inequívocas** de enunciados matemáticos y algoritmos, usando lógica precisa.

.UBAfiuba
FACULTAD DE INGENIERÍA

Algunos asistentes de pruebas

A continuación presentamos una lista de los principales, clasificados por sus fundamentos lógicos:

Algunos asistentes de pruebas

A continuación presentamos una lista de los principales, clasificados por sus fundamentos lógicos:

- **Teoría de conjuntos**: Isabelle/ZF, Metamath, Mizar
- **Teoría simple de tipos**: HOL4, HOL Light, Isabelle/HOL
- **Teoría dependiente de tipos**: Agda, Coq, Lean, Matita, PVS
- **Lógica de primer orden, de tipo Lisp**: ACL2

Algunos asistentes de pruebas

A continuación presentamos una lista de los principales, clasificados por sus fundamentos lógicos:

- **Teoría de conjuntos**: Isabelle/ZF, Metamath, Mizar
- **Teoría simple de tipos**: HOL4, HOL Light, Isabelle/HOL
- **Teoría dependiente de tipos**: Agda, Coq, Lean, Matita, PVS
- **Lógica de primer orden, de tipo Lisp**: ACL2

.UBAfiuba FACULTAD DE INGENIERÍA

Índice de contenidos

1 Introducción
- Blockchains, Criptomonedas y Smart contracts
- Cardano
- ACTUS
- Verificación formal de software

2 Escribiendo contratos ACTUS en Cardano
- Notación del estándar ACTUS
- Contratos en Cardano

3 Verificando propiedades en contratos en Marlowe
- El modelo de Marlowe
- Pruebas sencillas sobre contratos específicos
- Warnings en Auction

4 Posibles temas de desarrollo futuro

.UBAfiuba
FACULTAD DE INGENIERÍA

Índice de contenidos

1 Introducción
- Blockchains, Criptomonedas y Smart contracts
- Cardano
- ACTUS
- Verificación formal de software

2 Escribiendo contratos ACTUS en Cardano
- Notación del estándar ACTUS
- Contratos en Cardano

3 Verificando propiedades en contratos en Marlowe
- El modelo de Marlowe
- Pruebas sencillas sobre contratos específicos
- Warnings en Auction

4 Posibles temas de desarrollo futuro

.UBAfiuba FACULTAD DE INGENIERÍA

Notación

- **Atributos de contrato:** Representan los términos contractuales que definen el flujo de dinero en un contrato financiero.

Notación

- **Atributos de contrato:** Representan los términos contractuales que definen el flujo de dinero en un contrato financiero.
- **Starting date**: t_0 representa la fecha de comienzo del contrato, y marca el instante en el cual las condiciones y estado del contrato están siendo representados.

Notación

- **Atributos de contrato:** Representan los términos contractuales que definen el flujo de dinero en un contrato financiero.
- **Starting date**: t_0 representa la fecha de comienzo del contrato, y marca el instante en el cual las condiciones y estado del contrato están siendo representados.
- **Variables de estado**: Las variables de estado describen el estado de un contrato, para un tiempo determinado de su ciclo de vida. Algunos ejemplos de las mismas son:

Notación

- **Atributos de contrato:** Representan los términos contractuales que definen el flujo de dinero en un contrato financiero.
- **Starting date**: t_0 representa la fecha de comienzo del contrato, y marca el instante en el cual las condiciones y estado del contrato están siendo representados.
- **Variables de estado**: Las variables de estado describen el estado de un contrato, para un tiempo determinado de su ciclo de vida. Algunos ejemplos de las mismas son:
  - Notional Principal.
  - Nominal Interest Rate.

Notación

- **Atributos de contrato:** Representan los términos contractuales que definen el flujo de dinero en un contrato financiero.
- **Starting date**: t_0 representa la fecha de comienzo del contrato, y marca el instante en el cual las condiciones y estado del contrato están siendo representados.
- **Variables de estado**: Las variables de estado describen el estado de un contrato, para un tiempo determinado de su ciclo de vida. Algunos ejemplos de las mismas son:
  - Notional Principal.
  - Nominal Interest Rate.

  En general, el 'estado' representa ciertos términos de un contrato que pueden cambiar a lo largo de su ciclo de ejecución, de acuerdo a eventos programados o no programados.

- **Eventos**: Un evento de contrato e_t^k se refiere a cualquier evento *programado* o *no programado* en un momento determinado t y de un tipo determinado k.

Los eventos del contrato **marcan puntos específicos** en el tiempo en el que se **intercambian flujos de efectivo** o se **actualiza el estado del contrato**.

- **Funciones de transición de estado** (*State Transition Functions* o STF): definen la transición de las variables de estado desde el *pre-evento* hacia el *post-evento*, cuando un cierto evento e_t^k ocurre.
  Esto provoca que el *pre-evento* y *post-evento* reciban la notación de t^- y t^+ respectivamente.

- **Funciones de transición de estado** (*State Transition Functions* o STF): definen la transición de las variables de estado desde el *pre-evento* hacia el *post-evento*, cuando un cierto evento e_t^k ocurre.
  Esto provoca que el *pre-evento* y *post-evento* reciban la notación de t^- y t^+ respectivamente.

Estas funciones son específicas para un tipo de evento y contrato. Su notación es de la forma **STF_[event type]_[contract type]()**.

- **Funciones de transición de estado** (*State Transition Functions* o STF): definen la transición de las variables de estado desde el *pre-evento* hacia el *post-evento*, cuando un cierto evento e_t^k ocurre.
  Esto provoca que el *pre-evento* y *post-evento* reciban la notación de t^- y t^+ respectivamente.

Estas funciones son específicas para un tipo de evento y contrato. Su notación es de la forma **STF_[event type]_[contract type]()**.

Por ejemplo: La STF para un evento de tipo IP en el contrato PAM se escribe como STF_IP_PAM () y modifica (entre otras) a la variable **Ipac** desde el pre-evento $\mathbf{Ipac}_{t-}$ al post-evento $\mathbf{Ipac}_{t+}$.

.UBAfiuba FACULTAD DE INGENIERÍA

- **Funciones de pago** (*Payoff Functions* o POF): definen como el flujo de dinero $c \in \mathbb{R}$ ocurre para un determinado evento e_t^k.
  El mismo es obtenido del estado actual y los términos del contrato. Si fuera necesario, el flujo de dinero puede ser indexado con el tiempo del evento: c_t.

- **Funciones de pago** (*Payoff Functions* o POF): definen como el flujo de dinero $c \in \mathbb{R}$ ocurre para un determinado evento e_t^k.
  El mismo es obtenido del estado actual y los términos del contrato. Si fuera necesario, el flujo de dinero puede ser indexado con el tiempo del evento: c_t.

Las funciones de pago son específicas para un tipo de evento y contrato, y su notación es la siguiente:
POF_[event type]_[contract type] ().

- **Funciones de pago** (*Payoff Functions* o POF): definen como el flujo de dinero $c \in \mathbb{R}$ ocurre para un determinado evento e_t^k.
  El mismo es obtenido del estado actual y los términos del contrato. Si fuera necesario, el flujo de dinero puede ser indexado con el tiempo del evento: c_t.

Las funciones de pago son específicas para un tipo de evento y contrato, y su notación es la siguiente:
POF_[event type]_[contract type] ().

Por ejemplo: La POF para un evento de tipo IP en el contrato PAM se escribe como POF_IP_PAM().

Índice de contenidos

1 Introducción
- Blockchains, Criptomonedas y Smart contracts
- Cardano
- ACTUS
- Verificación formal de software

2 Escribiendo contratos ACTUS en Cardano
- Notación del estándar ACTUS
- Contratos en Cardano

3 Verificando propiedades en contratos en Marlowe
- El modelo de Marlowe
- Pruebas sencillas sobre contratos específicos
- Warnings en Auction

4 Posibles temas de desarrollo futuro

.UBAfiuba FACULTAD DE INGENIERÍA

Introducción

En esta sección describiré la estructura de 3 contratos ACTUS que escribí para la blockchain Cardano, bajo la supervisión de IOHK.

Introducción

En esta sección describiré la estructura de 3 contratos ACTUS que escribí para la blockchain Cardano, bajo la supervisión de IOHK.

Agregar contratos ACTUS en la blockchain Cardano implica agregar, para cada contrato:

Introducción

En esta sección describiré la estructura de 3 contratos ACTUS que escribí para la blockchain Cardano, bajo la supervisión de IOHK.

Agregar contratos ACTUS en la blockchain Cardano implica agregar, para cada contrato:

- Scheduling.

Introducción

En esta sección describiré la estructura de 3 contratos ACTUS que escribí para la blockchain Cardano, bajo la supervisión de IOHK.

Agregar contratos ACTUS en la blockchain Cardano implica agregar, para cada contrato:

- Scheduling.
- Inicialización de variables de estado.

Introducción

En esta sección describiré la estructura de 3 contratos ACTUS que escribí para la blockchain Cardano, bajo la supervisión de IOHK.

Agregar contratos ACTUS en la blockchain Cardano implica agregar, para cada contrato:

- Scheduling.
- Inicialización de variables de estado.
- Funciones de transición de estado y de pago, para cada evento relevante.

Introducción

En esta sección describiré la estructura de 3 contratos ACTUS que escribí para la blockchain Cardano, bajo la supervisión de IOHK.

Agregar contratos ACTUS en la blockchain Cardano implica agregar, para cada contrato:

- Scheduling.
- Inicialización de variables de estado.
- Funciones de transición de estado y de pago, para cada evento relevante.
- Validación mediante los tests propuestos por el estándar ACTUS.

Introducción

En esta sección describiré la estructura de 3 contratos ACTUS que escribí para la blockchain Cardano, bajo la supervisión de IOHK.

Agregar contratos ACTUS en la blockchain Cardano implica agregar, para cada contrato:

- Scheduling.
- Inicialización de variables de estado.
- Funciones de transición de estado y de pago, para cada evento relevante.
- Validación mediante los tests propuestos por el estándar ACTUS.

Dicho código fue incorporado en la rama principal del repositorio de marlowe-cardano. Por lo que usuarios podrán utilizar dichos contratos en el futuro.

Estructura del proyecto ACTUS en Cardano

A grandes rasgos, la estructura del generador de contratos ACTUS tiene los siguientes módulos:

- **Domain**: El mismo está conformado por los archivos que modelan el dominio de los contratos ACTUS.

  En este módulo podemos encontrar los tipos de **eventos**, **términos** y el **Estado** del contrato, entre otros.

Estructura del proyecto ACTUS en Cardano

A grandes rasgos, la estructura del generador de contratos ACTUS tiene los siguientes módulos:

- **Domain**: El mismo está conformado por los archivos que modelan el dominio de los contratos ACTUS.

  En este módulo podemos encontrar los tipos de **eventos**, **términos** y el **Estado** del contrato, entre otros.

- **Generator**: Se implementan los diferentes generadores y compatibilidad hacia el lenguaje Marlowe.

  Dichos generadores se utilizan para obtener código en Marlowe, Javascript, Haskell o Blockly, y pueden generar contratos estáticos (sin riesgos o eventos externos).

.UBAfiuba FACULTAD DE INGENIERÍA

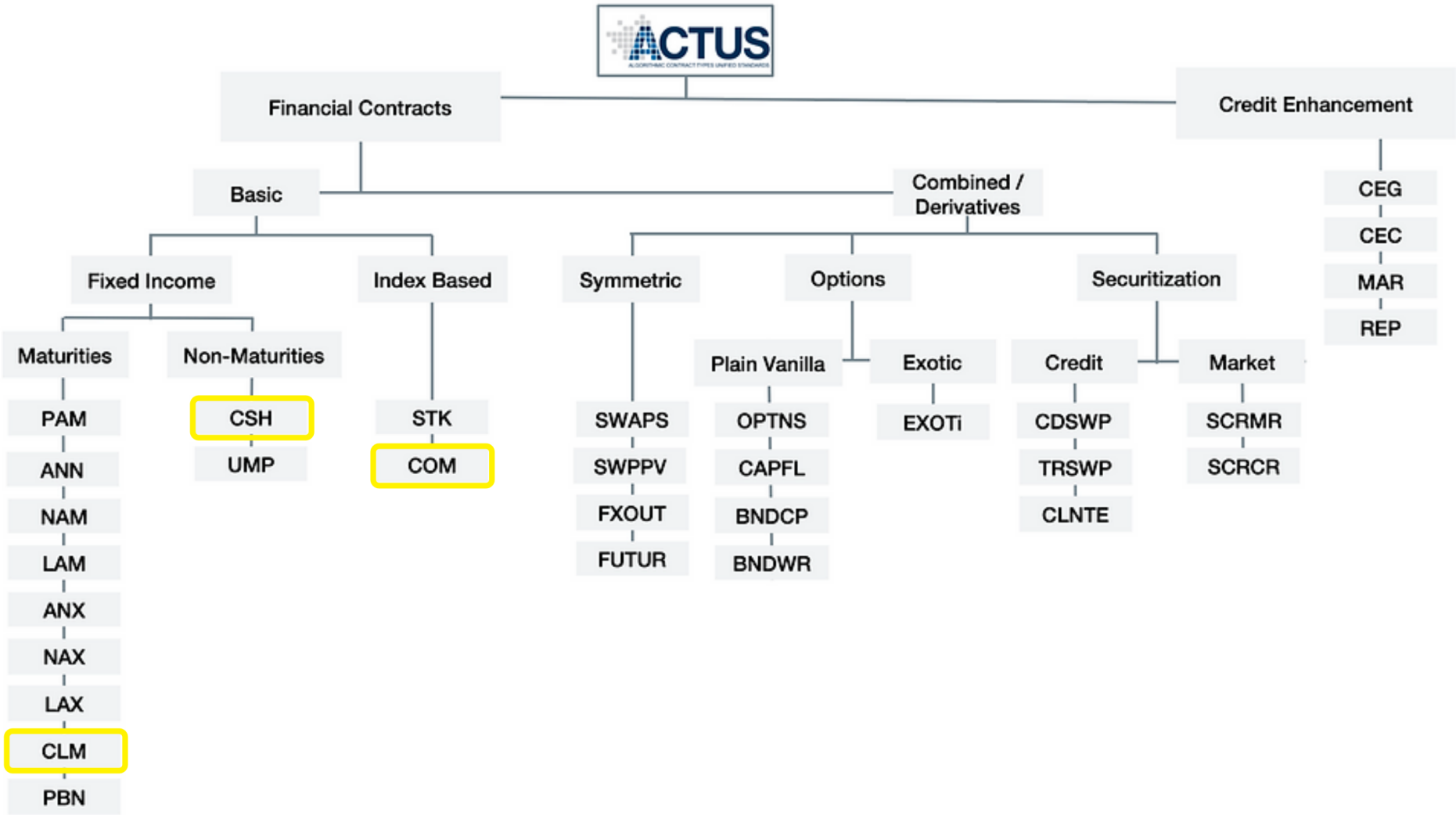
- **Model**: Se encuentra la lógica expuesta por el estándar ACTUS.

  Este módulo es responsable del **scheduling**, la **inicialización de variables de estado** y **funciones de transición de estado y de pago**.

- **Model**: Se encuentra la lógica expuesta por el estándar ACTUS.

  Este módulo es responsable del **scheduling**, la **inicialización de variables de estado** y **funciones de transición de estado y de pago**.

- **Utility**: Se encuentran algunos módulos con funciones que se utilizan para aislar la lógica del **cálculo y convenciones de fechas**, que suele tornarse complejo y repetitivo durante los contratos.



Contratos que implementé como parte del desarrollo de mi tesis

Algunos fragmentos del código necesario para poder implementar el *Scheduling* en el contrato CLM.

```
_SCHED_IP_CLM
  ContractTermsPoly
    { cycleAnchorDateOfInterestPayment = Just ipanx,
      cycleOfInterestPayment = Just ipcl,
      maturityDate = Just md,
      scheduleConfig
    } = generateRecurrentSchedule ipanx ipcl {includeEndDay = True}
    md scheduleConfig

_SCHED_IP_CLM
  ContractTermsPoly
    { cycleAnchorDateOfInterestPayment = Nothing,
      cycleOfInterestPayment = Just ipcl,
      maturityDate = Just md,
      initialExchangeDate = Just ied,
      scheduleConfig
    } = generateRecurrentSchedule (ied <+> ipcl) ipcl {includeEndDay
     = True} md scheduleConfig
```

Algunos ejemplos adicionales:

```
-- Cash (CSH) Monitoring STF
_STF_AD_CSH :: ContractStatePoly a b -> b -> ContractStatePoly a b
_STF_AD_CSH st@ContractStatePoly {} t =
  st
    { sd = t
    }
```

```
-- Call Money (CLM) POF's

_POF_IP_CLM :: ActusNum a => a -> a -> a -> a -> a -> a
_POF_IP_CLM o_rf_curs ipac ipnr nt y_sd_t = o_rf_curs * (ipac +
    y_sd_t * ipnr * nt)

_POF_IED_CLM :: RoleSignOps a => a -> CR -> a -> a
_POF_IED_CLM o_rf_curs cntrl nt = _zero - o_rf_curs * _r cntrl * nt
```

Índice de contenidos

1 Introducción
- Blockchains, Criptomonedas y Smart contracts
- Cardano
- ACTUS
- Verificación formal de software

2 Escribiendo contratos ACTUS en Cardano
- Notación del estándar ACTUS
- Contratos en Cardano

3 Verificando propiedades en contratos en Marlowe
- El modelo de Marlowe
- Pruebas sencillas sobre contratos específicos
- Warnings en Auction

4 Posibles temas de desarrollo futuro

.UBAfiuba FACULTAD DE INGENIERÍA

Índice de contenidos

1 Introducción
- Blockchains, Criptomonedas y Smart contracts
- Cardano
- ACTUS
- Verificación formal de software

2 Escribiendo contratos ACTUS en Cardano
- Notación del estándar ACTUS
- Contratos en Cardano

3 Verificando propiedades en contratos en Marlowe
- El modelo de Marlowe
- Pruebas sencillas sobre contratos específicos
- Warnings en Auction

4 Posibles temas de desarrollo futuro

.UBAfiuba FACULTAD DE INGENIERÍA

El modelo de Marlowe

Marlowe está diseñado para soportar la ejecución de contratos financieros en la blockchain, específicamente en Cardano.

El modelo de Marlowe

Marlowe está diseñado para soportar la ejecución de contratos financieros en la blockchain, específicamente en Cardano.

Antes de describir estos constructores, debemos analizar el enfoque general para **modelar contratos en Marlowe**, el contexto y las partes involucradas cuando se ejecutan los mismos.

- Contratos

Contratos

```
data Contract = Close
              | Pay party payee token value contract
              | If observation contract1 contract2
              | When [Case] timeout contract
              | Let valueId value contract
              | Assert observation contract
```

- Contratos

```
data Contract = Close
              | Pay party payee token value contract
              | If observation contract1 contract2
              | When [Case] timeout contract
              | Let valueId value contract
              | Assert observation contract
```

- Participantes y roles

- Contratos

```
data Contract = Close
              | Pay party payee token value contract
              | If observation contract1 contract2
              | When [Case] timeout contract
              | Let valueId value contract
              | Assert observation contract
```

- Participantes y roles
- Valores y *tokens*

- Contratos

```
data Contract = Close
              | Pay party payee token value contract
              | If observation contract1 contract2
              | When [Case] timeout contract
              | Let valueId value contract
              | Assert observation contract
```

- Participantes y roles
- Valores y *tokens*
- Cuentas

- Contratos

```
data Contract = Close
              | Pay party payee token value contract
              | If observation contract1 contract2
              | When [Case] timeout contract
              | Let valueId value contract
              | Assert observation contract
```

- Participantes y roles
- Valores y *tokens*
- Cuentas
- Pasos y Estados

- Contratos

```
data Contract = Close
              | Pay party payee token value contract
              | If observation contract1 contract2
              | When [Case] timeout contract
              | Let valueId value contract
              | Assert observation contract
```

- Participantes y roles
- Valores y *tokens*
- Cuentas
- Pasos y Estados
- Simulación omnisciente (*Marlowe Playground*)

Breve ejemplo de un contrato de Intercambio

```
When [Case (Deposit "alice" "alice" ada price)
        (When [ Case aliceChoice
                  (When [ Case bobChoice
                            (If (aliceChosen `ValueEQ` bobChosen)
                                 agreement
                                 arbitrate)
                          ]
                          60 -- Slot limite para la eleccion de Bob
                          arbitrate)
                ]
                40  -- Slot limite para la eleccion de Alice
                Close)
     ]
     10   -- Slot limite para el Deposito
     Close

-- agreement y arbitrate son contratos auxiliares que realizan los
    pagos o devoluciones correspondientes
```

Ejecución de un contrato Marlowe

Ejecutar un contrato de Marlowe en la cadena de bloques de Cardano significa restringir las transacciones generadas por el usuario a la lógica del contrato.

Ejecución de un contrato Marlowe

Ejecutar un contrato de Marlowe en la cadena de bloques de Cardano significa restringir las transacciones generadas por el usuario a la lógica del contrato.

Una **transacción** contiene una **lista ordenada de *entradas* o *acciones***. El intérprete de Marlowe se ejecuta durante la validación de transacciones.

Procesando una transacción

1. Se evalúa (o reduce) el contrato *paso* a *paso* hasta que **no se puede cambiar más sin procesar alguna entrada**, una condición que se llama *quiescent* (o inactiva).

   Esto implica reducir los `When` cuyos timeouts hayan sido superados, y todos los constructores `If`, `Let`, `Pay` y `Close`.

Procesando una transacción

1. Se evalúa (o reduce) el contrato *paso* a *paso* hasta que **no se puede cambiar más sin procesar alguna entrada**, una condición que se llama *quiescent* (o inactiva).

   Esto implica reducir los `When` cuyos timeouts hayan sido superados, y todos los constructores `If`, `Let`, `Pay` y `Close`.

2. Se procesa (o aplica) la primera entrada y luego el contrato se vuelve a procesar hasta la inactividad.

Procesando una transacción

1. Se evalúa (o reduce) el contrato *paso* a *paso* hasta que **no se puede cambiar más sin procesar alguna entrada**, una condición que se llama *quiescent* (o inactiva).

   Esto implica reducir los `When` cuyos timeouts hayan sido superados, y todos los constructores `If`, `Let`, `Pay` y `Close`.

2. Se procesa (o aplica) la primera entrada y luego el contrato se vuelve a procesar hasta la inactividad.

Este proceso se repite **hasta que se procesan todas las entradas**. En cada paso, **el contacto actual y el estado cambiarán** y es posible que se realicen pagos.

.UBAfiuba FACULTAD DE INGENIERÍA

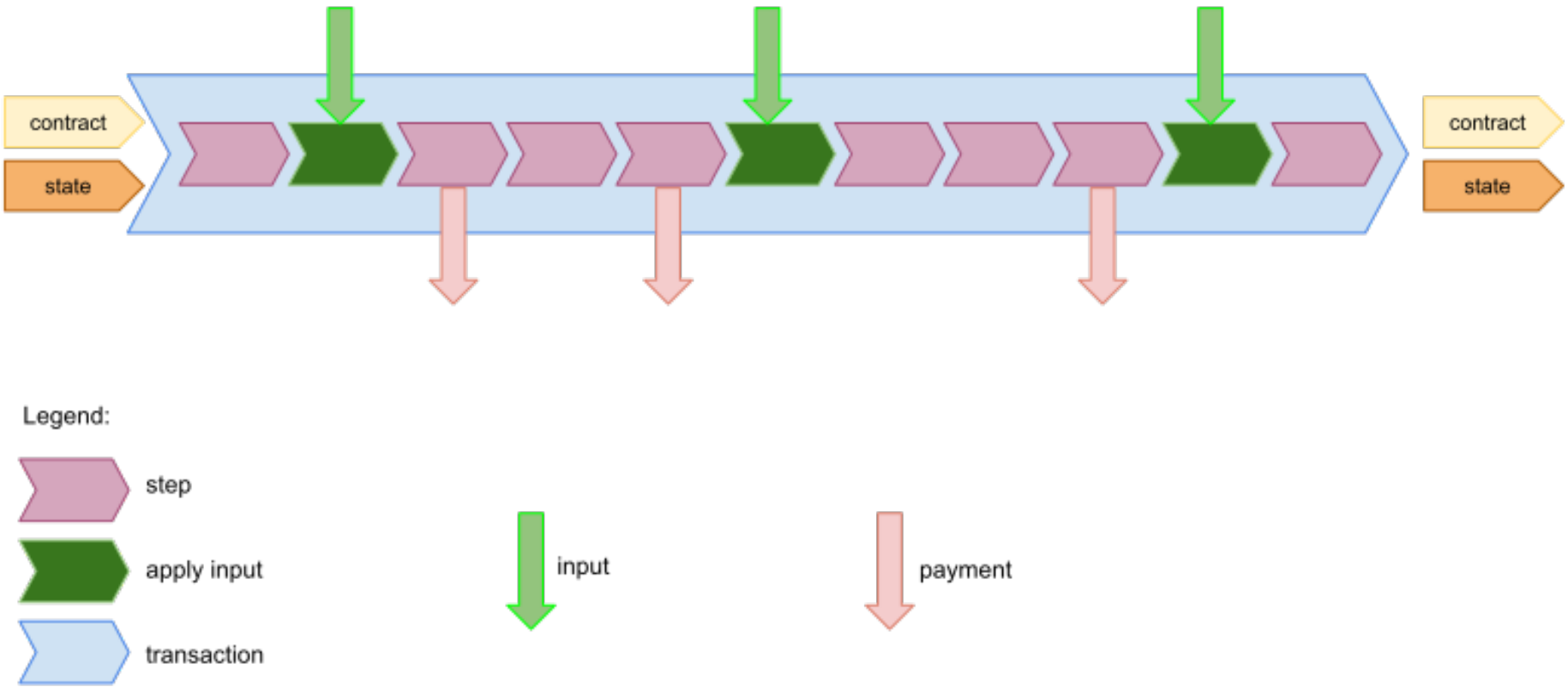
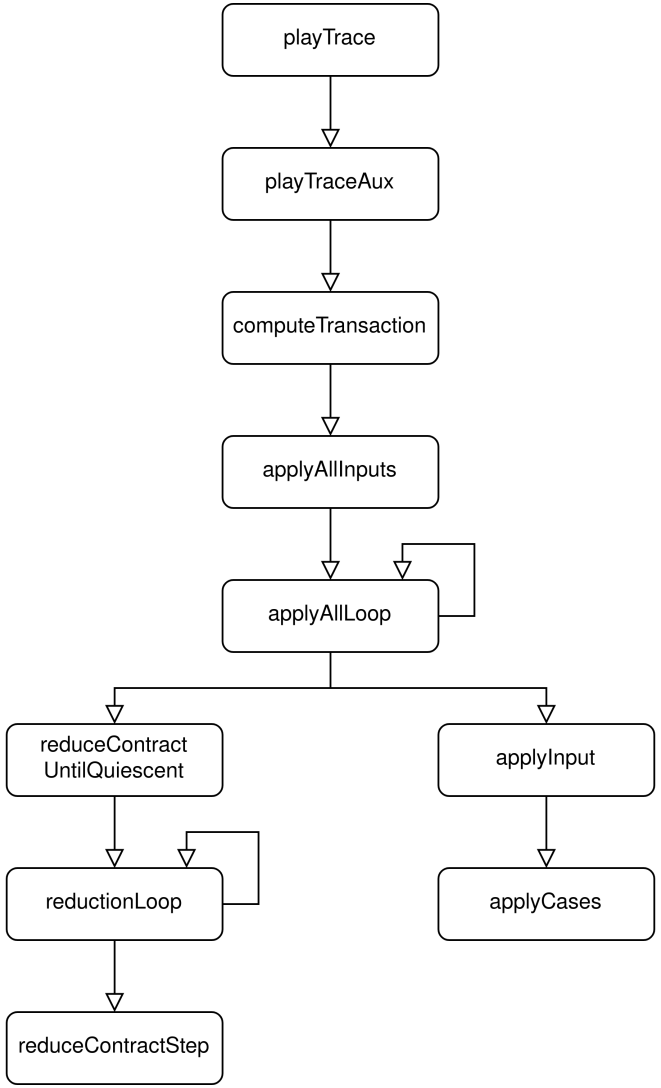
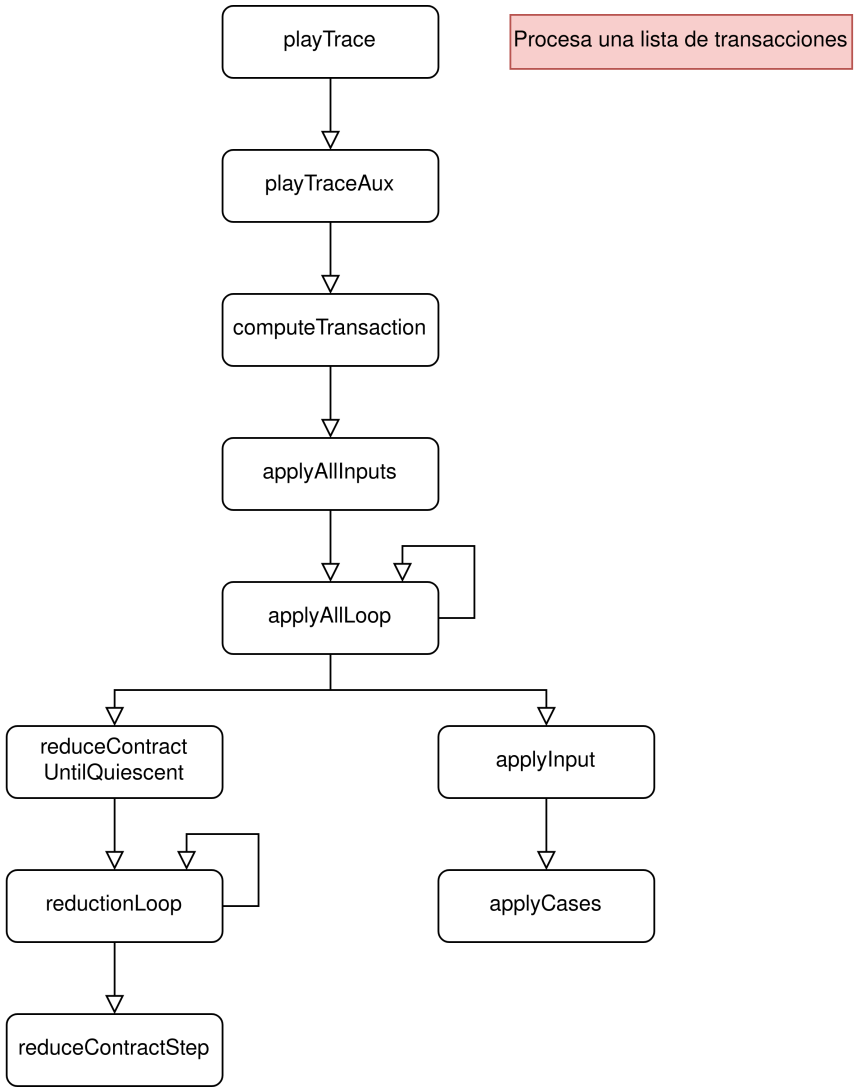


Diagrama de ejemplo de una transacción. Extraído de la documentación oficial del Modelo de Marlowe.

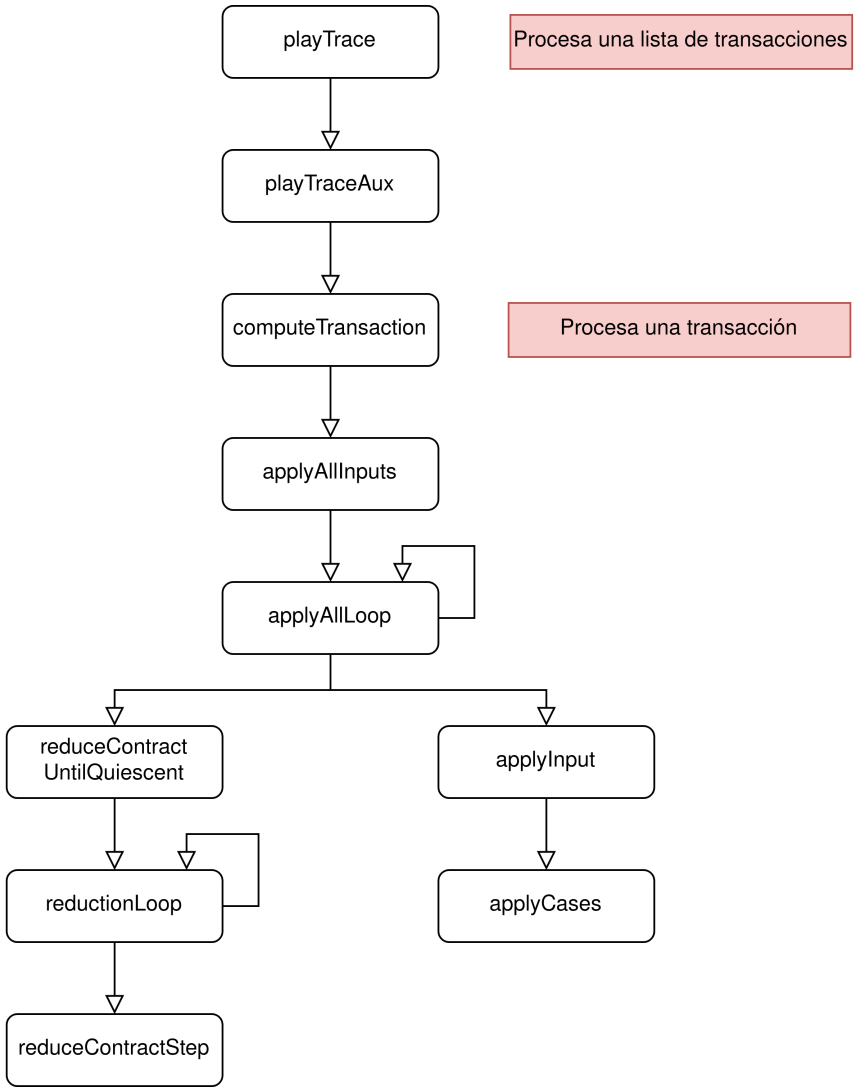
Funciones utilizadas en el modelo



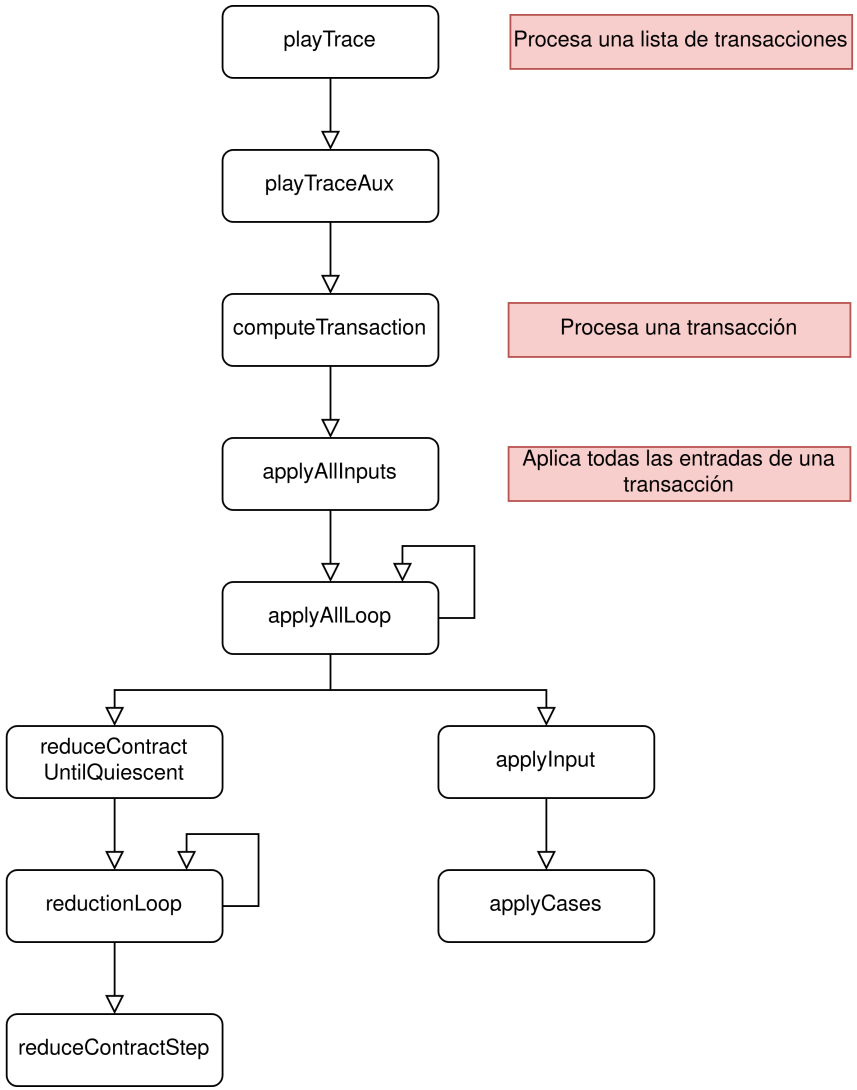
Funciones utilizadas en el modelo



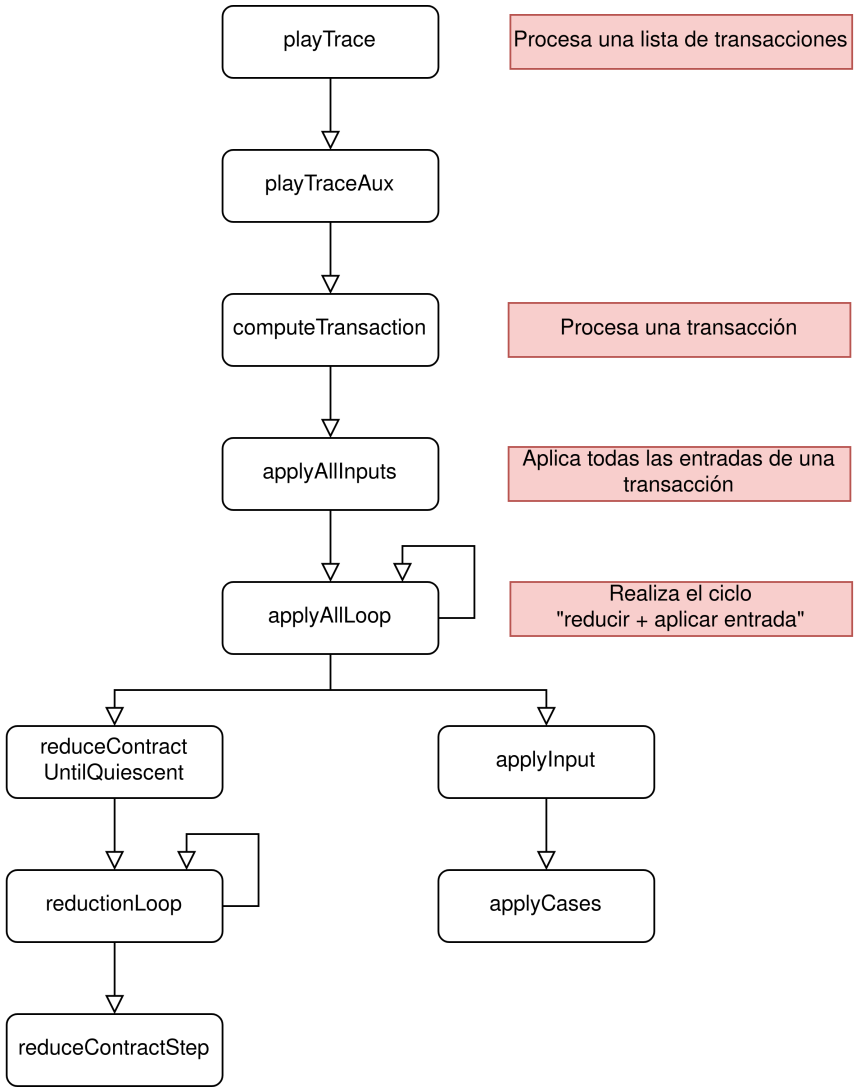
Funciones utilizadas en el modelo



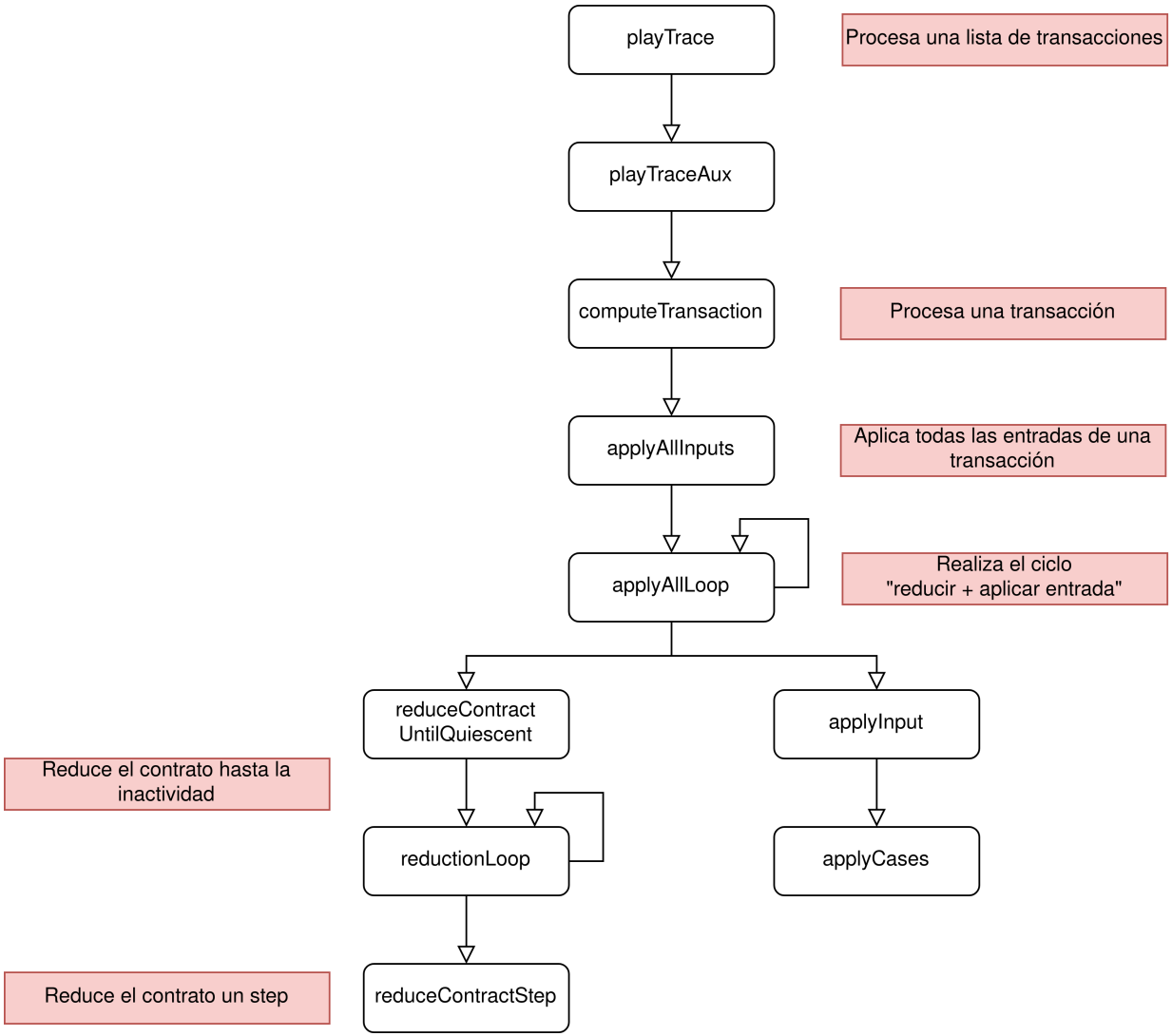
Funciones utilizadas en el modelo



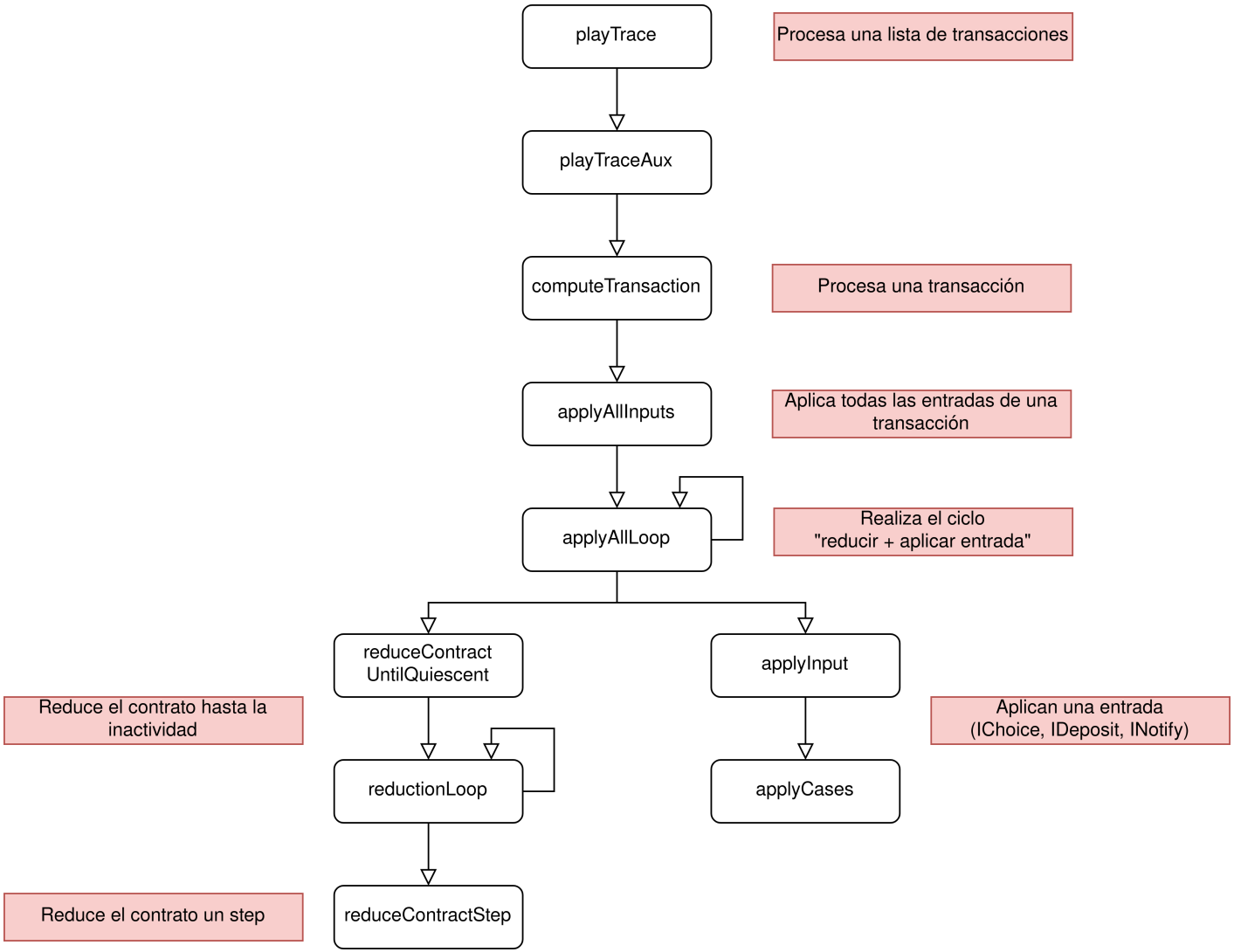
Funciones utilizadas en el modelo



Funciones utilizadas en el modelo



Funciones utilizadas en el modelo



Índice de contenidos

1 Introducción
- Blockchains, Criptomonedas y Smart contracts
- Cardano
- ACTUS
- Verificación formal de software

2 Escribiendo contratos ACTUS en Cardano
- Notación del estándar ACTUS
- Contratos en Cardano

3 Verificando propiedades en contratos en Marlowe
- El modelo de Marlowe
- Pruebas sencillas sobre contratos específicos
- Warnings en Auction

4 Posibles temas de desarrollo futuro

.UBAfiuba FACULTAD DE INGENIERÍA

Prueba de *slot* no decreciente en COM

Supongamos que queremos probar la siguiente propiedad:

> *"**Evaluar applyAllInputs en el contrato COM4**, para cualquier environment y state dado, **produce un nuevo estado con slot mayor al actual**."*

Este tipo de prueba evita la 'vuelta al pasado' por parte del contrato, que podrían llevar a caminos de ejecución no deseados.

Prueba del lema de *slot* para un contrato COM.

Un ejemplo de una propiedad que probé es la siguiente:

```
lemma applyAllInputsDoesNotDecreaseMinSlot :
"applyAllInputs env sta COM4 inputList =
    ApplyAllSuccess reduced warnings payments newstate cont ⟹
  (minSlot sta) ≤ (minSlot newstate)"
  apply auto
  apply (cases inputList)
  apply auto
  by (smt (z3) ApplyAllResult.distinct(1)
                 ApplyResult.case(1)
                 ApplyResult.case(2)
                 ApplyResult.exhaust
                 COM4_def
                 applyAllLoop_preserves_minSlot
                 applyCases_preserves_minSlot applyInput.simps(1))
```

.UBAfiuba FACULTAD DE INGENIERÍA

Índice de contenidos

1 Introducción
- Blockchains, Criptomonedas y Smart contracts
- Cardano
- ACTUS
- Verificación formal de software

2 Escribiendo contratos ACTUS en Cardano
- Notación del estándar ACTUS
- Contratos en Cardano

3 Verificando propiedades en contratos en Marlowe
- El modelo de Marlowe
- Pruebas sencillas sobre contratos específicos
- Warnings en Auction

4 Posibles temas de desarrollo futuro

.UBAfiuba FACULTAD DE INGENIERÍA

El contrato Auction

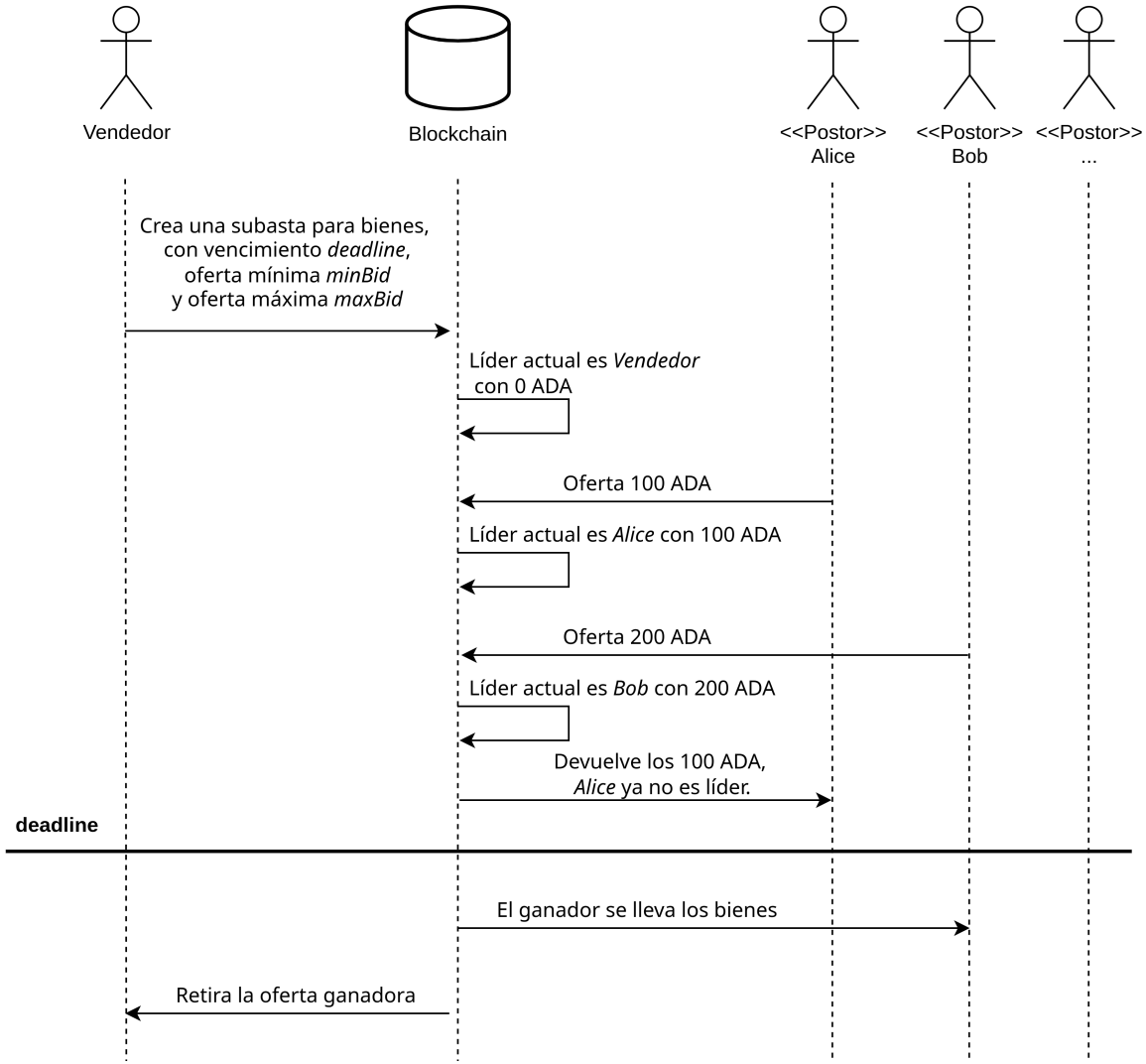
Una subasta (*Auction*) es una venta organizada en la cual el comprador (postor) que pague la mayor cantidad de dinero o de bienes a cambio del producto es el ganador de la misma.

El contrato Auction

Una subasta (*Auction*) es una venta organizada en la cual el comprador (postor) que pague la mayor cantidad de dinero o de bienes a cambio del producto es el ganador de la misma.

En particular, en una subasta basada en una *blockchain*, cada postor debe declarar el dinero primero para convertirse en el mejor postor. Los mismos recuperan su dinero cuando otro postor los supera.

Se puede ver en el siguiente diagrama de secuencia, el comportamiento esperado por la subasta:



Pruebas sobre el contrato *Auction*

Durante parte del desarrollo de mi tesis, probé que **cualquier contrato Auction no genera advertencias**.

Pruebas sobre el contrato *Auction*

Durante parte del desarrollo de mi tesis, probé que **cualquier contrato Auction no genera advertencias**.

Esto implicó:

- **Traducción** del contrato Auction en Haskell a Isabelle.

Pruebas sobre el contrato *Auction*

Durante parte del desarrollo de mi tesis, probé que **cualquier contrato Auction no genera advertencias**.

Esto implicó:

- **Traducción** del contrato Auction en Haskell a Isabelle.
- Prueba de la **terminación del generador** de contratos Auction.

Pruebas sobre el contrato *Auction*

Durante parte del desarrollo de mi tesis, probé que **cualquier contrato Auction no genera advertencias**.

Esto implicó:

- **Traducción** del contrato Auction en Haskell a Isabelle.
- Prueba de la **terminación del generador** de contratos Auction.
- Determinación de una **invariante** a partir de la cual ningún tipo de *warning* puede ocurrir.

Pruebas sobre el contrato *Auction*

Durante parte del desarrollo de mi tesis, probé que **cualquier contrato Auction no genera advertencias**.

Esto implicó:

- **Traducción** del contrato Auction en Haskell a Isabelle.
- Prueba de la **terminación del generador** de contratos Auction.
- Determinación de una **invariante** a partir de la cual ningún tipo de *warning* puede ocurrir.
- Prueba de **ausencia de Warnings** para todas las funciones involucradas en procesar una transacción.

Probando terminación del contrato *Auction*

Dado que Isabelle/HOL sigue una lógica de funciones totales, la terminación es un requerimiento fundamental.

Isabelle intenta probar la terminación automáticamente cuando se realiza la definición. Pero en algunas circunstancias **es posible que falle**, y la misma tiene que ser probada manualmente.

Probando terminación del contrato *Auction*

Dado que Isabelle/HOL sigue una lógica de funciones totales, la terminación es un requerimiento fundamental.

Isabelle intenta probar la terminación automáticamente cuando se realiza la definición. Pero en algunas circunstancias **es posible que falle**, y la misma tiene que ser probada manualmente.

La terminación no solo es una propiedad deseable en los contratos, sino que también le **permite a Isabelle y al usuario utilizar teoremas adicionales**, como por ejemplo el de inducción.

Métrica para la terminación

```
fun evalBoundAuction :: "(contractLoopType + (handleChooseType + handleDepositType
    )) ⇒ nat" where

"evalBoundAuction (Inl (_, ps, qs, _)) =
       2 * length ps + 4 * length qs + 1" |
"evalBoundAuction (Inr (Inl (_, ps, qs, _, p))) =
       2 * length ps + 4 * length qs + (if p ∈ set qs then 0 else 8)" |
"evalBoundAuction (Inr (Inr (_, ps, qs, _, p))) =
       2 * length ps + 4 * length qs + (if p ∈ set ps then 0 else 8)"
```

Invariante para la ausencia de warnings

Las acciones de los usuarios son imprevisibles y pueden generar *warnings* independientes a la semántica. Algunos ejemplos para este contrato podrían ser:

Invariante para la ausencia de warnings

Las acciones de los usuarios son imprevisibles y pueden generar *warnings* independientes a la semántica. Algunos ejemplos para este contrato podrían ser:

- Realizar más de una vez una elección (*ReduceShadowing*).

Invariante para la ausencia de warnings

Las acciones de los usuarios son imprevisibles y pueden generar *warnings* independientes a la semántica. Algunos ejemplos para este contrato podrían ser:

- Realizar más de una vez una elección (*ReduceShadowing*).
- Ofertar una cantidad negativa (*NonPositiveDeposit*).

Invariante para la ausencia de warnings

Las acciones de los usuarios son imprevisibles y pueden generar *warnings* independientes a la semántica. Algunos ejemplos para este contrato podrían ser:

- Realizar más de una vez una elección (*ReduceShadowing*).
- Ofertar una cantidad negativa (*NonPositiveDeposit*).
- Dinero insuficiente en la cuenta para la oferta realizada (*PartialPay*).

Invariante para la ausencia de warnings

Las acciones de los usuarios son imprevisibles y pueden generar *warnings* independientes a la semántica. Algunos ejemplos para este contrato podrían ser:

- Realizar más de una vez una elección (*ReduceShadowing*).
- Ofertar una cantidad negativa (*NonPositiveDeposit*).
- Dinero insuficiente en la cuenta para la oferta realizada (*PartialPay*).
- Cantidad en el *Pay* no sea positiva o las cuentas tengan suficiente dinero (*NonPositivePay* y *PartialPay*).

Invariante para la ausencia de warnings

Las acciones de los usuarios son imprevisibles y pueden generar *warnings* independientes a la semántica. Algunos ejemplos para este contrato podrían ser:

- Realizar más de una vez una elección (*ReduceShadowing*).
- Ofertar una cantidad negativa (*NonPositiveDeposit*).
- Dinero insuficiente en la cuenta para la oferta realizada (*PartialPay*).
- Cantidad en el *Pay* no sea positiva o las cuentas tengan suficiente dinero (*NonPositivePay* y *PartialPay*).
- Que el *minBid* del contrato sea positivo, sino se producirá alguna combinación de los *warnings* anteriores.

.UBAfiuba FACULTAD DE INGENIERÍA

Definición en Isabelle de la invariante para *Auction*

Es por eso que se formularon una serie de **pre-condiciones o invariantes que deben ser satisfechas** para poder asegurar que el contrato no genera *warnings*.

```
definition invariantHoldsForAuction :: "AuctionTerms ⇒ AuctionWinner ⇒ Party
    list ⇒ Party list ⇒ State ⇒ bool" where

"invariantHoldsForAuction terms m ps qs curState =
      ((∀ x . x ∈ set qs ⟶
              ¬ member (partyToValueId x) (boundValues curState))
    ∧ (∀ x . x ∈ set ps ⟶
              findWithDefault 0 (partyToValueId x) (boundValues curState) > 0)
    ∧ (∀ x y . m = Some (x, y) ⟶
            ((lookup (y, token_ada) (accounts curState) =
              lookup (partyToValueId y) (boundValues curState))
          ∧ (findWithDefault 0 (partyToValueId y) (boundValues curState) > 0)
          ∧ (UseValue (partyToValueId y)) = x))
    ∧ (minBid terms > 0))"
```

Ausencia de warnings en Isabelle al computar una transacción

```
lemma auctionIsSafe_computeTransaction :
    "invariantHoldsForAuction terms m ps qs sta ⟹
     computeTransaction tra sta (contractLoop m ps qs terms) =
       TransactionOutput trec ⟹ txOutWarnings trec = []"

  using fixingIntervalPreservesInvariant auctionIsSafe_computeTransactionFixSta
  by (smt (verit, ccfv_SIG) IntervalResult.simps(6) closeIsSafe computeTransaction
       .simps fixInterval.elims)
```

Esta prueba depende de las demostraciones previas sobre todas las funciones en las que depende (*ApplyAllInputs*, *ReductionLoop*, *ReduceContractStep*, *ApplyInput*, etc.)

Índice de contenidos

1 Introducción
- Blockchains, Criptomonedas y Smart contracts
- Cardano
- ACTUS
- Verificación formal de software

2 Escribiendo contratos ACTUS en Cardano
- Notación del estándar ACTUS
- Contratos en Cardano

3 Verificando propiedades en contratos en Marlowe
- El modelo de Marlowe
- Pruebas sencillas sobre contratos específicos
- Warnings en Auction

4 Posibles temas de desarrollo futuro

.UBAfiuba FACULTAD DE INGENIERÍA

Posibles temas de desarrollo e investigación

- Con respecto a la escritura de contratos ACTUS en Cardano, actualmente **no se han terminado de escribir todos los contratos especificados por el estándar**.

Posibles temas de desarrollo e investigación

- Con respecto a la escritura de contratos ACTUS en Cardano, actualmente **no se han terminado de escribir todos los contratos especificados por el estándar**.
- En cuanto a la verificación de programas, es posible **tomar algunos contratos** (por ejemplo entre los ejemplos del *Playground*) **en los que no se esperan warnings y verificarlos** mediante Isabelle.

  Dichas verificaciones podrán beneficiarse de los lemas y técnicas utilizadas para los contratos *Auction* y *Close*.

Posibles temas de desarrollo e investigación

- Con respecto a la escritura de contratos ACTUS en Cardano, actualmente **no se han terminado de escribir todos los contratos especificados por el estándar**.
- En cuanto a la verificación de programas, es posible **tomar algunos contratos** (por ejemplo entre los ejemplos del *Playground*) **en los que no se esperan warnings y verificarlos** mediante Isabelle.

  Dichas verificaciones podrán beneficiarse de los lemas y técnicas utilizadas para los contratos *Auction* y *Close*.
- La **traducción automática de algunas sentencias de código Haskell a Isabelle**. Parte de esta idea es abarcada en [Torrini et al., 2007], aunque manteniendo la traducción manual.

¿Preguntas?