



Universidad Nacional de Rosario
Facultad de Ciencias Exactas,
Ingeniería y Agrimensura
Departamento de Ciencias de la
Computación



ANÁLISIS DE LENGUAJES DE PROGRAMACIÓN

Trabajo Práctico Final

Un EDSL para representar Bonos

Poma Lucas

Febrero 2024

1. Descripción del proyecto

1.1. Introducción del Dominio

El siguiente trabajo consiste en un EDSL cuyo objetivo es la representación de bonos y la posibilidad de realizar cálculos con los mismos, bajo ciertas suposiciones.

Para lograr un mejor entendimiento del trabajo primero haremos algunas definiciones.

Un bono es un instrumento financiero que emite una empresa (bonos corporativos/obligaciones negociables) o un estado (bonos soberanos), con el objetivo de financiarse, y que devuelve el capital recibido, con intereses, en un plazo determinado.

Existen diversas formas que pueden tomar los bonos. El bono mas simple de entender es el zero coupon bond (bono de cupón cero, zcb), es aquel que en una única fecha de vencimiento, restablece el capital en su totalidad y paga los intereses correspondientes. Tenemos básicamente 4 elementos que conforman este pago, que en si mismo es un bono:

- **Fecha**
- Monto correspondiente a la **renta** (de los intereses)
- Monto correspondiente a la **amortización** (del capital)
- **Moneda** en la que se efectúa el pago

Otra categorización de los bonos son los **bullet bonds**, bonos que tienen pagos de intereses periódicos (o ninguno, en caso de un zcb) y que solo reintegran la totalidad del capital en la fecha de vencimiento. Es decir tienen amortización 0 hasta la fecha de vencimiento donde restablecen el 100 % del capital.

Como última categoría relevante podemos nombrar los **amortization bonds**, bonos que incluyen en los pagos, pago de intereses (renta) y amortización de parte del capital. Es importante entender que la renta generada, si los intereses sobre el capital se mantienen al mismo nivel, ira disminuyendo, ya que el capital va disminuyendo conforme al pago de amortización.

También, es posible ligar los valores de los pagarés del bono con diferentes medidas. Las más conocidas en Argentina son, el tipo de cambio del peso al dólar (bonos dólar linked) y la inflación (bonos CER), ambas medidas son emitidas por el banco central. Por ejemplo, en el caso de un bono CER, si un pago es de \$100, y cuando el bono fue emitido, la medida de CER emitida por el banco central era de 1.0, entonces, si en la fecha de pago constatamos que el valor del CER es de 1.5 (es decir 50 % de inflación), el bono pagaría en esa fecha \$150.

Otro concepto interesante es que los bonos luego de ser licitados, entran en lo que se denomina mercado secundario, donde pueden ser comprados o vendidos en el mercado y su precio depende exclusivamente de la oferta y la demanda del mismo. Esto es relevante, ya que el **precio de mercado** difiere del valor de los pagos del bono, ya que se tienen en cuenta otros factores, como la posibilidad de incumplimiento del pago por parte del emisor.

1.2. Introducción Técnica

El lenguaje permite combinar bonos más simples, cuya definición es más sencilla, en bonos más complejos, para posteriormente poder realizar cálculos sobre los mismos. La idea principal es poder ver el flujo de fondos de un bono (una tabla que muestra de manera ordenada los pagos en cada fecha, conocido como **cash flow**), y también, poder obtener algunos valores teóricos y fechas relevantes del bono.

Para estos cálculos, podemos agregar una lista de suposiciones, como la fecha, el precio de mercado, el valor CER, el valor del tipo de cambio y la cantidad de bonos.

Podemos definir un bono y almacenarlo en una variable, para posteriormente poder combinar estas variables en un nuevo bono de posiblemente mayor complejidad.

Se permite definir un **portafolio**, básicamente un conjunto de bonos ya definidos y sus cantidades, que denotarían una posible posición personal de los mismos. Sobre este portafolio definido podemos también calcular el flujo de fondos, con lo cual, podríamos ver los pagos esperados de todas nuestras posesiones.

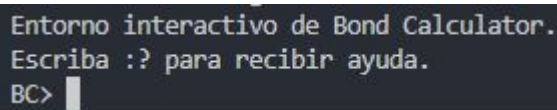
El lenguaje posee azúcar sintáctico para facilitar la definición de ciertos bonos. También permite el uso de diversos escaladores que afectan el valor de los pagos.

Permitimos hacer estas definiciones de bonos y portafolios en un archivo para luego cargarlos en el entorno interactivo (o podemos directamente definirlos en este) para, posiblemente, realizar cálculos.

2. Instalación

Teniendo instalados Stack y Haskell, y contando con el código fuente del proyecto, deberemos situarnos con la terminal en la raíz del proyecto.

Una vez hecho esto, solo resta ejecutar **stack run** y obtendremos el siguiente entorno interactivo.



```
Entorno interactivo de Bond Calculator.  
Escriba :? para recibir ayuda.  
BC> |
```

Figura 1: Entorno Interactivo

3. Manual de Uso

3.1. Definiciones

El programa nos permite definir tanto bonos como portafolios y guardarlos en variables para luego poder acceder a estas definiciones llamando a la variable indicada.

3.1.1. Archivos de Definiciones

Estos archivos, que contienen sólo definiciones de bonos y portafolios, permiten cargarlas al entorno interactivo. Deben obligatoriamente utilizar la extensión **".bnd"**

3.1.2. Definición de Bonos

Procederemos a mostrar una gramática simplificada para facilitar el entendimiento de la definición de bonos.

$$\begin{aligned}
def & ::= 'def' \text{ VAR } '=' \text{ bond} \\
bond & ::= \text{VAR} \\
& \quad | \text{bond } '\&' \text{ bond} \\
& \quad | '(' \text{ bond } ')' \\
& \quad | 'scale' \text{ scaler bond} \\
& \quad | 'at' \text{ date payment} \\
& \quad | \text{iterate payment} \\
& \quad | \text{iterate } 'interest' \text{ PERCENT amortpay } 'of' \text{ DECIMAL CURRENCY} \\
& \quad | \text{iterate amortpay } 'interest' \text{ PERCENT } 'of' \text{ DECIMAL CURRENCY} \\
payment & ::= \text{rentpay amortpay CURRENCY} \\
& \quad | \text{amortpay rentpay CURRENCY} \\
& \quad | 'zero' \\
rentpay & ::= 'rent' \text{ DECIMAL} \\
& \quad | 'rent' \text{ PERCENT } 'of' \text{ DECIMAL} \\
& \quad | \epsilon \\
amortpay & ::= 'amort' \text{ DECIMAL} \\
& \quad | 'amort' \text{ PERCENT } 'of' \text{ DECIMAL} \\
& \quad | \epsilon \\
scaler & ::= \text{DECIMAL} \\
& \quad | 'CER' \text{ DECIMAL} \\
& \quad | 'DL' \text{ DECIMAL} \\
iterate & ::= 'repeat' \text{ INTEGER freq date} \\
date & ::= \text{INTEGER } '/' \text{ INTEGER } '/' \text{ INTEGER} \\
freq & ::= 'ANNUAL' / 'SEMIANNUAL' / 'QUARTERLY' / 'MONTHLY'
\end{aligned}$$

Donde:

- *VAR* denota una cadena alfanumérica.
- *CURRENCY* denota una cadena alfabética precedida por el símbolo '\$'.
- *INTEGER* denota una cadena numérica.
- *DECIMAL* denota 2 *INTEGER* utilizando '/' como separación.
- *PERCENT* denota un *DECIMAL* sucedido por un '%'.
- ϵ denota la cadena vacía.

3.1.3. Definición de Portafolios

Procederemos a mostrar una gramática simplificada para facilitar el entendimiento de la definición de portafolios.

$$\begin{aligned} port &::= 'portfolio' VAR '=' '[' vars ']' \\ vars &::= INTEGER VAR ',' vars \\ &\quad | \quad INTEGER VAR \end{aligned}$$

Donde:

- *VAR* denota una cadena alfanumérica.
- *INTEGER* denota una cadena numérica.

3.2. Entorno Interactivo

Siguiendo los pasos de mencionados en la parte de instalación, obtendremos el entorno interactivo del proyecto. Desde allí, podemos lanzar comandos, realizar definiciones y llamar a operaciones sobre definiciones.

3.2.1. Comandos

Disponemos de 4 comandos.

- **:browse**, permite ver los bonos y portafolios cargados en el entorno global hasta el momento
- **:load *file***, permite cargar el archivo de definiciones *file* al entorno
- **:quit**, permite salir del entorno interactivo, terminando su ejecución
- **:help**, muestra la lista de comandos disponibles

3.2.2. Expresiones

Procederemos a mostrar una gramática simplificada para facilitar el entendimiento de las expresiones que podemos formar

```

exp ::= 'print' supps bond
      | 'dates' supps bond
      | 'values' supps bond
      | 'detail' supps bond
      | 'cashflow' supps bond
      | 'portcashflow' supps VAR

supps ::= 'suppose' '[' conds ']'
        | ε

conds ::= cond ',' conds
        | cond

cond ::= 'BCUSD' DECIMAL
        | 'BCCER' DECIMAL
        | 'TODAY'
        | 'MARKET' DECIMAL CURRENCY
        | 'DATE' date
        | 'QUANTITY' INTEGER

```

Donde:

- *VAR* denota una cadena alfanumérica.
- *CURRENCY* denota una cadena alfabética precedida por el símbolo '\$'.
- *INTEGER* denota una cadena numérica.
- *DECIMAL* denota 2 *INTEGER* utilizando '.' como separación.
- *date* fue definido en 3.1.2
- *bond* fue definido en 3.1.2

3.2.3. Operaciones

Disponemos de 6 operaciones. En todas el parámetro opcional *supps* denota las condiciones supuestas con las cuales se evaluará el bono/portafolio.

- **print (*supps*) *bond***, imprime en pantalla el bono *bond* en forma de lista de tuplas
- **dates (*supps*) *bond***, imprime en pantalla datos relacionados con fechas relevantes del bono *bond*:
 - **Supposed Date**, nos indica sobre que fecha nos estamos situando para los cálculos (por defecto sera 01/01/1900)
 - **Maturity Date**, nos indica la fecha de vencimiento del bono (último pago)
 - **Last Coupon**, nos indica la fecha del último pago antes de la fecha supuesta.
 - **Next Coupon**, nos indica la fecha del próximo pago desde la fecha supuesta.

- **Days to Next Coupon**, nos indica la diferencia de días entre el ítem anterior y la fecha supuesta.
- **Remaining Payments**, nos indica la cantidad de pagos que quedan hasta que finalice el bono.
- **values (*supps*) bond**, imprime en pantalla datos relacionados con valores teóricos del bono *bond*:
 - **Supposed Price**, nos indica sobre que precio de mercado estamos suponiendo para los cálculos (por defecto no habrá ningún valor)
 - **Nominal Value**, nos indica el valor del capital por el cual fue emitido el bono (suma de todas las amortizaciones)
 - **Residual Value**, nos indica el valor del capital restante desde la fecha supuesta (suma de las amortizaciones restantes)
 - **Accrued Interest**, nos indica los intereses corridos, es decir el monto que correspondería de la próxima renta desde la anterior fecha de pago hasta la supuesta (un monto proporcional de la siguiente renta a cobrar por los días transcurridos entre ambas fechas)
 - **Technical Value**, suma del valor residual y los intereses corridos
 - **Parity**, relación porcentual entre el precio de mercado y el valor técnico.
- **detail (*supps*) bond**, imprime en pantalla ambas operaciones anteriores (valores y dates) sobre *bond*
- **cashflow (*supps*) bond**, imprime en pantalla el flujo de fondos del bono *bond* desde la fecha supuesta.
Se indican, en orden, la **fecha** del pago, la **moneda** del pago, el monto correspondiente a la **renta**, el monto correspondiente a la **amortización**, la suma de estos últimos 2 como el **total** a recibir por el pago, los **escaladores** del pago que no pudieron ser aplicados en la evaluación.
- **portcashflow (*supps*) port**, imprime en pantalla el flujo de fondos de todos los bonos del portafolio *port*, escalados por sus cantidades, desde la fecha supuesta.

3.2.4. Suposiciones

Como comentamos anteriormente, podemos suponer condiciones para condicionar las operaciones, utilizando la palabra clave *suppose* y pasándole una lista separada por ", " de las condiciones. Disponemos de 6 condiciones:

- **BCUSD d** para suponer con el decimal d el valor del tipo de cambio de ARS a USD, emitido por el banco central.
- **BCCER d** para suponer con el decimal d el valor de la medida CER, emitida por el banco central.
- **TODAY** para suponer el día de la fecha en el sistema.
- **MARKET d c** para suponer el precio de mercado como el decimal d en moneda c .

- **DATE** d para suponer la fecha d .
- **QUANTITY** i para suponer la cantidad entera i del bono a evaluar

4. Ejemplos Prácticos

Para dejar mas en claro el uso, mostramos una serie de ejemplos de la realidad. Estos ejemplos están definidos en `arg.bnd` en la carpeta de `examples`. Una vez en el entorno interactivo escribimos en la terminal `:l examples/arg.bnd` para cargar las definiciones en el entorno.

4.1. AL30

El AL30 es uno de los bonos mas conocidos de Argentina, y resulta interesante por su complejidad de definirlo.

Su definición extraída de <https://www.cohen.com.ar/Bursatil/Especie/AL30>

- Amortización: **13** cuotas **semestrales**, primera del **4 %** y las demás del **8 %** del capital que es de **1 USD** por bono, desde el 9/7/2024
- Intereses (**anualizados**):
 - Primera fecha de pago de intereses: **9/7/2021**
 - 4/9/2020 hasta 9/7/2021: **0.125 %**
 - 9/7/2021 hasta 9/7/2023: **0.5 %**
 - 9/7/2023 hasta 9/7/2027: **0.75 %**
 - 9/7/2027 hasta vencimiento 9/7/2030: **1.75 %**

Procedemos a definirlo en `arg.bnd` de la siguiente manera:

```
def al30p1 = at 9/7/2021 rent 0.125% of 1.0 $USD
def al30p2 = repeat 4 SEMIANNUAL 9/1/2022 rent 0.25% of 1.0 $USD
def al30p3 = at 9/1/2024 rent 0.375% of 1.0 $USD
def al30p4 = repeat 1 SEMIANNUAL 9/7/2024 interest 0.75% amort 0.04 of 1.0 $USD
def al30p5 = repeat 6 SEMIANNUAL 9/1/2025 interest 0.75% amort 0.08 of 0.96 $USD
def al30p6 = repeat 6 SEMIANNUAL 9/1/2028 interest 1.75% amort 0.08 of 0.48 $USD

def al30 = al30p1 & al30p2 & al30p3 & al30p4 & al30p5 & al30p6
```

- `al30p1`: El primer pago de intereses es luego de pasar el semestre, por lo tanto, corresponde el interés anualizado directamente. Como solo es 1 cuota con estas características, lo definimos directamente.
- `al30p2`: Como solo es pago de renta, simplemente iteramos para repetir el pago, **4** pagos con frecuencia semestral desde el **9/1/2022**, el interés anual es de **0.5 %**, por ello, cada pago será de **0.25 %**
- `al30p3`: Solo esta cuota tiene estas características, todavía no hay amortización y es la primera con interés anual de **0.75 %**, corresponde el interés semestral de **0.375 %**

- al30p4: Primera y única cuota con amortización de 4 %, podría haberse definido directamente, pero por fines didácticos la definimos con la sintaxis que refleja un **amortization bond**.

Realizamos una única iteración, con interés anualizado de **0.75 %** y una amortización de **0.04**

- al30p5: Las siguientes **6** cuotas mantienen el interés, pero la amortización pasa a ser de **0.08**, y ahora el capital base restante es de $1.0 - 0.04 = 0.96$.

Utilizamos la misma sintaxis que en el anterior, que tiene en cuenta la disminución del capital a medida que se amortiza para el cálculo del monto correspondiente a renta.

- al30p6: Ultimas **6** cuotas donde se eleva el interés a **1.75 %** anual, y se mantiene la misma amortización de **0.08** pero ahora restan **0.48** del capital.

Teniendo todas estas partes definidas (que a su vez, son bonos por si solas), podemos entonces combinarlas y formar un bono mas complejo, en este caso, el AL30.

Realizamos las operaciones detail y cashflow, para posteriormente comprobar su correctitud en paginas externas. Utilizamos las suposiciones correspondientes, la fecha del día utilizando **TODAY**, los bonos se muestran en el mercado con el precio cada 100 unidades, por ello utilizamos **QUANTITY 100** y colocamos el precio de mercado correspondiente para estos 100, **44.8 USD**.

Cargamos arg.bnd, realizamos las operaciones y obtenemos los siguientes resultados, viendo que los valores coinciden:

Descripción	Valor
ticker	AL30D
Precio	44.8
Variación diaria	↑2.99%
Paridad	44.8%
Valor Técnico	100.1

Figura 2: Valores Teóricos AL30 extraído de <https://bonistas.com/bonos-argentinos/AL30>

BC> detail suppose [TODAY, QUANTITY 100, MARKET 44.8 \$USD] al30			
Full Detail for Bond al30			
Supposed Date	29/02/2024	Supposed Price	44.800 USD
Maturity Date	09/07/2030	Nominal Value	100.000 USD
Last Coupon	09/01/2024	Residual Value	100.000 USD
Next Coupon	09/07/2024	Accrued Interest	0.105 USD
Days to Next Coupon	131	Technical Value	100.105 USD
Remaining Payments	13	Parity	44.753 %

Figura 3: Valores Teóricos AL30 obtenidos de la evaluación

FECHA PAGO	VR (%)	VR CARTERA (%)	RENTA (R)	AMORTIZACIÓN (A)	A + R
09/07/2024	100,00	100,00%	0,3750	4,0000	4,3750
09/01/2025	96,00	96,00%	0,3600	8,0000	8,3600
09/07/2025	88,00	88,00%	0,3300	8,0000	8,3300
09/01/2026	80,00	80,00%	0,3000	8,0000	8,3000
09/07/2026	72,00	72,00%	0,2700	8,0000	8,2700
09/01/2027	64,00	64,00%	0,2400	8,0000	8,2400
09/07/2027	56,00	56,00%	0,2100	8,0000	8,2100
09/01/2028	48,00	48,00%	0,4200	8,0000	8,4200
09/07/2028	40,00	40,00%	0,3500	8,0000	8,3500
09/01/2029	32,00	32,00%	0,2800	8,0000	8,2800
09/07/2029	24,00	24,00%	0,2100	8,0000	8,2100
09/01/2030	16,00	16,00%	0,1400	8,0000	8,1400
09/07/2030	8,00	8,00%	0,0700	8,0000	8,0700

Figura 4: Flujo de Fondos AL30 extraídos de <https://www.cohen.com.ar/Bursatil/Especie/AL30>

BC> cashflow suppose [TODAY, QUANTITY 100] al30					
Cashflow for Bond al30					
Date	Currency	Rent	Amort	Total	Scalars
09/07/2024	USD	0.375	4.000	4.375	N/A
09/01/2025	USD	0.360	8.000	8.360	N/A
09/07/2025	USD	0.330	8.000	8.330	N/A
09/01/2026	USD	0.300	8.000	8.300	N/A
09/07/2026	USD	0.270	8.000	8.270	N/A
09/01/2027	USD	0.240	8.000	8.240	N/A
09/07/2027	USD	0.210	8.000	8.210	N/A
09/01/2028	USD	0.420	8.000	8.420	N/A
09/07/2028	USD	0.350	8.000	8.350	N/A
09/01/2029	USD	0.280	8.000	8.280	N/A
09/07/2029	USD	0.210	8.000	8.210	N/A
09/01/2030	USD	0.140	8.000	8.140	N/A
09/07/2030	USD	0.070	8.000	8.070	N/A

Figura 5: Flujo de Fondos AL30 obtenidos de la evaluación

4.2. TV24

Otro bono bastante operado en Argentina es el TV24, un bono Dolar Linked. Su definición extraída de <https://www.cohen.com.ar/Bursatil/Especie/TV24>

- Amortización: Integra al vencimiento
- Intereses (**anualizados**):
 - Primera fecha de pago de intereses: **30/10/2022**
 - hasta vencimiento 30/4/2024: **0.4 %**

Procedemos a definirlo en arg.bnd de manera simple:

```
def tv24 = scale DL 152.83 repeat 4 SEMIANNUAL
  30/10/2022 interest 0.4% of 152.83 $ARS
```

Definimos la parte interna del bono de esta manera, que resulta azúcar sintáctico para definir un **bullet bond** ya que la amortización es solo al vencimiento y mientras tanto solo paga renta por los intereses, que es de un **0.4 %** anualizado.

Constatamos que el tipo de cambio al día de la emisión 18/04/2022 en la página del BCRA es de **152.83**, así que lo colocamos en el escalador. Cada bono se emite por el equivalente a 1 USD por lo tanto colocamos el equivalente en ARS como capital.

Primero lo evaluamos para ver como queda conformado sin suponer un tipo de cambio ni fecha, vemos que los escaladores figuran en la tabla. Luego utilizamos las suposiciones correspondientes, la fecha del día utilizando **TODAY**, cantidad **100** y el tipo de cambio para el día emitido por el BCRA **841.15**, vemos que se aplica el escalador correspondiente.

```
BC> cashflow suppose [QUANTITY 100] tv24
```

Cashflow for Bond tv24					
Date	Currency	Rent	Amort	Total	Scalars
30/10/2022	ARS	30.566	0.000	30.566	DolarLinked 152.83
30/04/2023	ARS	30.566	0.000	30.566	DolarLinked 152.83
30/10/2023	ARS	30.566	0.000	30.566	DolarLinked 152.83
30/04/2024	ARS	30.566	15283.000	15313.566	DolarLinked 152.83

Figura 6: Flujo de Fondos TV24 sin suposiciones

```
BC> cashflow suppose [TODAY, QUANTITY 100, BCUSD 841.15] tv24
```

Cashflow for Bond tv24					
Date	Currency	Rent	Amort	Total	Scalars
30/04/2024	ARS	168.230	84115.000	84283.230	N/A

Figura 7: Flujo de Fondos TV24 obtenidos de la evaluación con suposiciones

Flujo de fondos teórico ⓘ

FECHA DE PAGO ↕	TOTAL ↕	AMORTIZACIÓN ↕	RENTA ↕	VALOR RESIDUAL ↕
30/04/2024	ARS 84.283,23	ARS 84.115,00	ARS 168,23	100,00 %

Figura 8: Flujo de Fondos TV24 extraídos de <https://trading.portfoliopersonal.com/Cotizaciones/Item/837352?plazo=3>

4.3. TX26

Por ultimo, un bono interesante que esta linkeado a la inflación es el bono CER TX26.

Su definición extraída de <https://www.cohen.com.ar/Bursatil/Especie/TX26>

- Amortización: **5** cuotas **semestrales**, **20 %** del capital que es de **1 ARS** por bono, desde el **9/11/2024**
- Intereses (**anualizados**):
 - Primera fecha de pago de intereses: **9/5/2021**
 - hasta vencimiento 9/11/2026: **2 %**

Procedemos a definirlo en arg.bnd de la siguiente manera:

```
def tx26p1 = repeat 7 SEMIANNUAL 9/5/2021 rent 1.0% of 1.0 $ARS
def tx26p2 = repeat 5 SEMIANNUAL 9/11/2024 interest 2.0% amort 20.0%
  of 1.0 of 1.0 $ARS
```

```
def tx26 = scale CER 22.544 tx26p1 & tx26p2
```

- tx26p1: Como solo es pago de renta, simplemente iteramos para repetir el pago, **7** pagos con frecuencia semestral desde el **9/5/2021**, el interés anual es de **2 %**, por ello, cada pago será de **1 %**
- tx26p2: Utilizamos la sintaxis que replica un **amortization bond**, con **5** cuotas, comenzando el **9/11/2024** con un interés anualizado de **2 %** y donde la amortización de cada cuota es un **20 %** del capital total que es de **1 ARS**

Teniendo ambas partes definidas las combinamos mediante '&' y lo que resta es escalarlas por un factor CER.

Constatamos que el CER al día 21/08/2020 (se toma el valor del CER informado en 10 días hábiles anteriores al día de emisión) en la página del BCRA es de **22.544**, así que lo colocamos en el escalador.

Primero lo evaluamos para ver como queda conformado sin suponer un valor de CER ni fecha, vemos que los escaladores figuran en la tabla. Luego utilizamos las suposiciones correspondientes, la fecha del día utilizando **TODAY**, cantidad **100** y el CER para el día **15/2/2024** (10 días hábiles anteriores al 27/2/2024, la fecha del día) emitido por el BCRA **246.02**.

BC> cashflow suppose [QUANTITY 100] tx26
Cashflow for Bond tx26

Date	Currency	Rent	Amort	Total	Scalars
09/05/2021	ARS	1.000	0.000	1.000	CER 22.544
09/11/2021	ARS	1.000	0.000	1.000	CER 22.544
09/05/2022	ARS	1.000	0.000	1.000	CER 22.544
09/11/2022	ARS	1.000	0.000	1.000	CER 22.544
09/05/2023	ARS	1.000	0.000	1.000	CER 22.544
09/11/2023	ARS	1.000	0.000	1.000	CER 22.544
09/05/2024	ARS	1.000	0.000	1.000	CER 22.544
09/11/2024	ARS	1.000	20.000	21.000	CER 22.544
09/05/2025	ARS	0.800	20.000	20.800	CER 22.544
09/11/2025	ARS	0.600	20.000	20.600	CER 22.544
09/05/2026	ARS	0.400	20.000	20.400	CER 22.544
09/11/2026	ARS	0.200	20.000	20.200	CER 22.544

Figura 9: Flujo de Fondos TX26 sin suposiciones

BC> cashflow suppose [TODAY, BCCER 246.02, QUANTITY 100] tx26
Cashflow for Bond tx26

Date	Currency	Rent	Amort	Total	Scalars
09/05/2024	ARS	10.913	0.000	10.913	N/A
09/11/2024	ARS	10.913	218.258	229.171	N/A
09/05/2025	ARS	8.730	218.258	226.988	N/A
09/11/2025	ARS	6.548	218.258	224.805	N/A
09/05/2026	ARS	4.365	218.258	222.623	N/A
09/11/2026	ARS	2.183	218.258	220.440	N/A

Figura 10: Flujo de Fondos TX26 obtenidos de la evaluación con suposiciones

FECHA PAGO	VR (%)	VR CARTERA (%)	RENTA (R)	AMORTIZACIÓN (A)	A + R
09/05/2024	100,00	100,00%	10,9129	0,0000	10,9129
09/11/2024	100,00	100,00%	10,9129	218,2577	229,1706
09/05/2025	80,00	80,00%	8,7303	218,2577	226,9880
09/11/2025	60,00	60,00%	6,5477	218,2577	224,8054
09/05/2026	40,00	40,00%	4,3652	218,2577	222,6229
09/11/2026	20,00	20,00%	2,1826	218,2577	220,4403

Figura 11: Flujo de Fondos TX26 extraídos de <https://www.cohen.com.ar/Bursatil/Especie/TX26>

4.4. Portafolio

Teniendo algunos bonos definidos podemos ahora mostrar la funcionalidad de crear un portafolio. Procedemos a definir un portafolio llamado `argy` de la siguiente manera:

```
portfolio argy = [5 al30, 1 tv24, 3 tx26]
```

Luego utilizando la operación `portcashflow` y suponiendo el día de la fecha, multiplicando por 100 todas las tenencias (supondríamos entonces que en nuestra cartera ficticia tendríamos 500 `al30`, 100 `tv24`, 300 `tx26`), y las medidas anteriormente nombradas emitidas por el BCRA, obtenemos el flujo de fondos esperado para toda nuestra cartera, teniendo en cuenta la cantidad de cada activo.

```
BC> :l examples/arg.bnd
Abriendo examples/arg.bnd...
BC> portcashflow suppose [TODAY, QUANTITY 100, BCCER 246.02, BCUSD 841.15] argy
Cashflow for portfolio argy
```

Date	Ticker	Currency	Rent	Amort	Total	Scalars
30/04/2024	tv24	ARS	168.230	84115.000	84283.230	N/A
09/05/2024	tx26	ARS	32.739	0.000	32.739	N/A
09/07/2024	al30	USD	1.875	20.000	21.875	N/A
09/11/2024	tx26	ARS	32.739	654.773	687.512	N/A
09/01/2025	al30	USD	1.800	40.000	41.800	N/A
09/05/2025	tx26	ARS	26.191	654.773	680.964	N/A
09/07/2025	al30	USD	1.650	40.000	41.650	N/A
09/11/2025	tx26	ARS	19.643	654.773	674.416	N/A
09/01/2026	al30	USD	1.500	40.000	41.500	N/A
09/05/2026	tx26	ARS	13.095	654.773	667.868	N/A
09/07/2026	al30	USD	1.350	40.000	41.350	N/A
09/11/2026	tx26	ARS	6.548	654.773	661.321	N/A
09/01/2027	al30	USD	1.200	40.000	41.200	N/A
09/07/2027	al30	USD	1.050	40.000	41.050	N/A
09/01/2028	al30	USD	2.100	40.000	42.100	N/A
09/07/2028	al30	USD	1.750	40.000	41.750	N/A
09/01/2029	al30	USD	1.400	40.000	41.400	N/A
09/07/2029	al30	USD	1.050	40.000	41.050	N/A
09/01/2030	al30	USD	0.700	40.000	40.700	N/A
09/07/2030	al30	USD	0.350	40.000	40.350	N/A

Figura 12: Flujo de Fondos del Portafolio

5. Organización de Archivos

Este proyecto consta de 7 módulos, el parser (.y utilizando Happy) y por último el modulo Main. Este ultimo, se encuentra en *app*, mientras que los demás archivos se encuentran en *src*. También contamos con una carpeta *examples* donde tenemos archivos de ejemplo .bnd para poder testear el programa.

- *Bond.hs* Contiene el AST del lenguaje básico y algunas funciones para trabajar bonos como listas de tuplas.
- *Errors.hs* Definición del tipo Error
- *Eval.hs* Contiene la función de evaluación principal, donde se realizan los cálculos necesarios y se aplican las condiciones supuestas.
- *MonadBnd.hs* Contiene la monada encargada de llevar el entorno, manejar errores y permitir imprimir en pantalla.

- *PrettyPrinter.hs* Se encarga de mostrar en pantalla de forma mas legible y elegante los resultados de la evaluación.
- *State.hs* Definición del estado que lleva la monada
- *Sugar.hs* Contiene el AST del lenguaje con azúcar sintáctico, las funciones para convertir al lenguaje básico y las definiciones de expresiones.
- *Main.hs* Flujo del programa ejecutable con el entorno interactivo.
- *Parse.y* Definición del parser del lenguaje en Yacc.

6. Decisiones de Diseño

6.1. Idea General

Para estructurar el programa decidí que iba a tener un árbol base con las operaciones primitivas y un árbol con las expresiones con azúcar sintáctico añadido.

Para pasar del árbol con sugar al base se utiliza una función llamada *convert*. Parseamos las definiciones como sugar y las guardamos en el entorno como un bono formado sólo con primitivas. Para evaluar una expresión también convertimos el bono con azúcar a un bono base y luego aplicamos las condiciones si es que las hay.

Por último para algunos cálculos y el pretty print, convertimos el árbol a una lista de tuplas que indican cada pago del bono y utilizando un paquete llamado Boxes formamos las tablas correspondientes para imprimirlas en pantalla.

6.2. Estructuras Utilizadas

```
import Data.Time.Calendar (Day)

type Var = String
type Currency = String

data Scaler = Mult Double | CER Double | DolarLinked Double
    deriving Show

data Payment = PZero | Pay Double Double Currency
    deriving Show

data Bond =
    And Bond Bond
  | Scale Scaler Bond
  | At Day Payment
    deriving Show
```

Este es el árbol base para la definición de un bono en base a primitivas, se encuentra en el modulo Bond. Denota en orden, combinar 2 bonos, escalar un bono, definir un pago en una determinada fecha.

En los pagos el primer valor double denota el monto correspondiente a la **renta**, el otro a la **amortización**.

```

data Frequency = Annual | SemiAnnual | Quarterly | Monthly
    deriving (Eq, Show)

type Iterator = (Int, Frequency, Day)

data SugarBond =
    SVar Var
  | SAnd SugarBond SugarBond
  | SScale Scaler SugarBond
  | SAt Day Payment
  | SRepeat Iterator Payment
  | SCoupon Iterator Double Double Money
    deriving Show

```

Este es el árbol que utilizamos al momento de parsear la entrada, este cuenta con azúcar sintáctico, luego al momento de la evaluación o al ingresar al entorno se lo convierte en el árbol base que mencionábamos anteriormente.

- SVar: El bono del entorno definido bajo el nombre de la variable
- SAnd: Combinar 2 bonos
- SScale: Escalar un bono
- SAt: Define un pago en una cierta fecha
- SRepeat: Repite un pago una cantidad de veces, con una frecuencia determinada, desde cierto día
- SCoupon: Azúcar sintáctico para definir un **amortization bond**, donde si la amortización es 0 (segundo double), entonces se trata de un **bullet bond**. En el parámetro de tipo Money está definido el capital base y la moneda.

```

data Cond = BCCER Double | BCUSD Double | Date Day | Market Money | ←
    Quantity Int | Today
    deriving Show

data DefOrExp = Def Var SugarBond | Eval Exp | Portfolio Var [(Int, Var)]
    deriving Show

data Exp =
    Print [Cond] SugarBond
  | Dates [Cond] SugarBond
  | Values [Cond] SugarBond
  | Detail [Cond] SugarBond
  | Cashflow [Cond] SugarBond
  | PortCashflow [Cond] Var
    deriving Show

```

Con esta estructura denotamos expresiones para que luego sean evaluadas bajo una posible lista de condiciones.


```
type Def = (Var, Bond)
type Port = (Var, [(Int, Var)])

data State = State
{
  env :: [Def], -- Entorno con variables globales y su valor
  currentPrice :: Maybe Money,
  currentDate :: Day,
  portfolios :: [Port]
}
```

Definimos el estado de esta manera donde:

- env: Denota el entorno donde definimos bonos bajo un nombre
- currentPrice: Se utiliza para guardar el precio de mercado supuesto para posiblemente utilizarlo en cálculos
- currentDate: Se utiliza para guardar la fecha supuesta para posiblemente utilizarla en cálculos
- portfolios: Denota el entorno donde definimos portafolios (lista de bonos y sus cantidades) bajo un nombre

7. Comentarios

Algo que me molestaba cuando ingresaba a los flujos de fondo de algunos bonos en páginas externas, era que, para bonos Dolar Linked o bonos CER, simplemente se mostraba el valor actual ya escalado, en lugar de figurar algo que indique que el bono esta siendo escalado a alguna medida. Por eso decidí incluir los escaladores en el flujo de fondos.

Por otro lado, decidí permitir un mismo bono pueda incluir pagos en distintas monedas y también que pueda ser escalado por varias medidas. Esto trae como contrapartida que algunos cálculos no tienen tanto sentido, como por ejemplo intentar calcular la paridad, ya que el precio de mercado esta en sólo una moneda.

A futuro, mediante modelos matemáticos y probabilísticos, podría ser posible, extender el proyecto para lograr calcular otros valores teóricos sobre los bonos y su valor presente neto.

8. Bibliografía

Referencias y Bibliografía

- [1] Composing contracts: an adventure in financial engineering, Simon Peyton Jones. URL: <https://www.microsoft.com/en-us/research/wp-content/uploads/2016/07/contracts-icfp.pdf>
- [2] Text.PrettyPrint.Boxes. URL: <https://hackage.haskell.org/package/boxes-0.1.5/docs/Text-PrettyPrint-Boxes.html>

- [3] Data.Time.Calendar. URL: <https://hackage.haskell.org/package/time-1.12.2/docs/Data-Time-Calendar.html#v:showList>
- [4] UNR, LCC 2022, ALP TP3
- [5] UNR, LCC 2022, ALP TP4
- [6] UNR, LCC 2023, Compiladores