me

Universidad de Córdoba

Verificación de la inferencia de tipos en el lenguaje de programación `Crystal`

Presentado ante la Facultad de Matemática, Astronomía, Física y Computación, como trabajo de grado

Por: Ramiro Barraco
Director: Dr. Soldevila Raffa, Mallku Ernesto
Director Representante: Dr. Ziliani, Beta

25 de agosto de 2022

Crystal Lang

’Facts, facts, facts,’ cries the scientist if he wants to emphasize the necessity of a firm foundation for science. What is a fact? A fact is a thought that is true. But the scientist will surely not recognize something which depends on men’s varying states of mind to be the firm foundation of science.

– Gottlob Frege (1956). "The thought: A logical inquiry"

Resumen

Crystal es un lenguaje de programación orientado a objetos, diseñado y desarrollado por Ary Borenszweig, Juan Wajnerman, Brian Cardiff y más de 300 colaboradores. Con una sintaxis inspirada en el lenguaje Ruby, es un lenguaje compilado con verificación estática de tipos, pero sin la necesidad de especificar tipos de variables o los argumentos de los métodos. Los tipos se resuelven mediante un algoritmo de inferencia de tipos globales. Crystal presentó su versión 1.0.0 este año y sigue en desarrollo con una comunidad sumamente activa, con más de 6000 Shards (librerias de Crystal).

En este trabajo intentaremos formalizar una pequeña parte del lenguaje, en particular la inferencia de tipos. Utilizaremos PLT Redex para modelar Crystal-Clear y la probaremos con una suite de test de nuestra autoría. También usaremos herramientas que Redex ofrece para Random-testing y probaremos propiedades del lenguaje como seguridad.

Formalizar esta semántica puede otorgarle mayor seguridad y confianza a los proyectos desarrollados en Crystal

Abstract

Crystal is a general purpose object oriented programming language, designed and developed by Ary Borenszweig, Juan Wajnerman, Brian Cardiff and more than 300 collaborators. With a syntax inspired in Ruby, is a compiled language with static type-checking, but specifying the types of variables or method arguments is generally unneeded. Types are resolved by an advanced global type inference algorithm. Crystal is currently in active development. It is released as free and open-source software. This year Crystal launched it's 1.0.0 version and has an active community with more than 6000 shards (Crystal libraries).

In this work we will try to formalize a little part of the language, in particular it's type inference. We will model Crystal-Clear using PLT redex and test it with a test suite of our own authorship. Also we will use Redex tools like Random-testing to prove language properties as security.

Formalize this semantic may give more security and confidence to projects developed in Crystal.

Índice general

Índice de figuras

Índice de cuadros

Introducción 1

1.1. Objetivos

En esta tesis se trabaja con la intención de definir un lenguaje formal llamado Crystal-Clear que modele parte de Crystal. En especial buscamos definir algunas funciones básicas en cualquier lenguaje de programación y luego concentrarnos en el algoritmo de inferencia de tipos. Para especificar un lenguaje de programación primero es importante entender que lo que se busca es describir la sintaxis y semántica del lenguaje , esto se logra a través de la utilización de diferentes herramientas que se verán a lo largo de este trabajo.

Otro objetivo de esta tesis es testear dicha formalización y convencernos de que Crystal-Clear realmente modela la parte del lenguaje que decidimos trabajar. Una vez estemos seguros de esto podremos utilizar herramientas como redex-check para probar que esté inferidor cumple con la propiedad safety(definida en pierce).

1.2. Motivación

Crystal es un lenguaje de programación que fue lanzado en 2021, como tal todavía tiene un largo camino por recorrer en temas como comunidad y desarrollo. Para ello un paso importante es conseguir una formalización completa de su lenguaje. En especial el inferidor de tipos que agrega nuevas cualidades no observadas en otros lenguajes ya trabajados.

1.3. Herramientas

En el trabajo de esta tesis se utilizaron múltiples herramientas tanto prácticas como teóricas. En la parte teórica utilizamos conceptos como gramáticas, (TODO). En cuanto a herramientas prácticas empleamos un lenguaje de programación

llamado Racket basado en Lisp. La utilización de esta herramienta es debido a la existencia de su librería redex, que nos permite definir estos objetos teóricos y usar las mismas para probar propiedades y desarrollar tests para convencernos de los resultados obtenidos.

1.4. Estructura de la tesis

La tesis estará dividida en 6 capítulos.

En el primero hablaremos de Crystal y Racket/Redex, Explicaremos propiedades interesantes de ambos lenguajes y se intentará transmitir un entendimiento acerca de las sintaxis de ambos.

En el segundo se explicarán conceptos teóricos utilizados que servirán para entender el trabajo de la tesis.

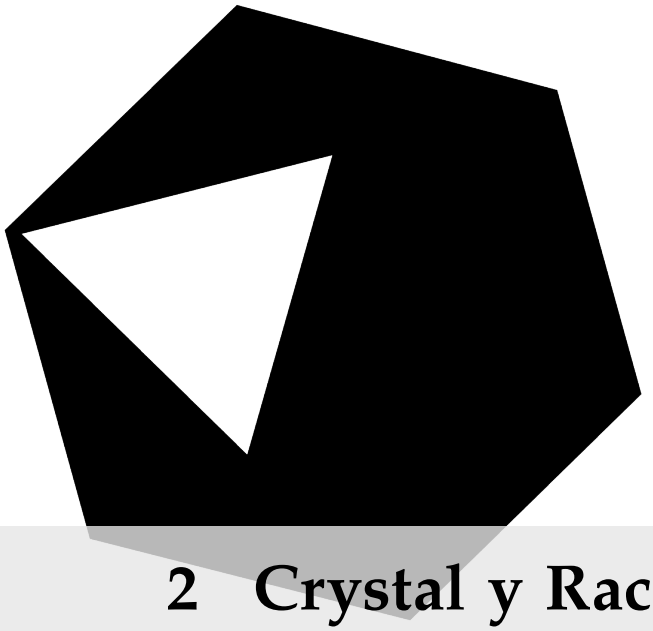
En el tercero se hablará de la gramática y semántica de Crystal-Clear nuestra formalización.

Luego seguiremos a la mecanización donde intentaremos comparar nuestro lenguaje a Crystal, observaremos resultados obtenidos y hablaremos de su utilidad para la comunidad, también comentaremos sobre carencias de la documentación.

En el penúltimo capítulos comentaremos trabajos relacionados, ya sea de otras formalizaciones de otros lenguajes como de conceptos teóricos similares desarrollados por otros investigadores.

Por último daremos a conocer nuestras conclusiones y posibles trabajos futuros.

Conceptos importantes y herramientas



2 Crystal y Racket

2.1. Resumen de cualidades de Crystal

Algunas cualidades importantes de nombrar de crystal son:

Es compilado esto le permite hacer un chequeo de tipos previo al tiempo de ejecucion y ofrece una mayor a velocidad.

Su sintaxis esta basada fuertemente en otro lenguaje llamado Ruby

A pesar de tener tipado estatico no es necesario definir los tipos de las variables, debido a su inferenciador de tipos. Esto lo logra atra vez de union types y type narrowing

Union type y Type narrowing

En crystal las variables o expresiones pueden consistir de multiples tipos. A esto lo llamamos union types, por ejemplo.

```
  if 1 + 2 == 3
  x = 1
else
  x = "hello"
end

x : Int32 | String
```

En este caso x tiene dos posibles tipos Int32 en el caso que x = 1 o String en caso de que x = "hello". Esto genera un problema al momento de evaluar ciertas expresiones, tengamos en consideracion ahora que usamos la variable x de la siguiente forma

```
  x = x + 3
```

Dependiendo del valor de x dicha expresion no puede ser evaluada, para esto se agrega un concepto llamado type narrowing. Es posible hacer un chequeo previo del tipo de esta forma

```
if x.is_a?(Int32)
  x = x + 3
end
```

Crystal nos asegura que el valor de x luego del if es un Int32 y lo cual nos permite que la expresion pueda ser evaluada. Es importante aclarar que los union type son asignados en tiempo de compilacion y que el chequeo de tipos que nos permite hacer type narrowing ocurre en la ejecucion.

2.2. ¿Que es racket/Redex y porque lo usamos?

Racket es un lenguaje basado en lisp y descendiente de scheme. Esta diseñado como una plataforma para implementar y desarollar lenguajes de programacion. Esta siendo desarollado por el grupo PLT fundado por Matthias Felleisen a mediados de los 90. Actualmente funcin como un projecto para crear un ambiente de programacion pedagogico.

Racket incluye distintas herramientas como macros, modulos, llamadas a la cola, cerraduras lexicas, continuaciones delimitadas, contratos, green threads, Os threads. Aunque la cualidad que mas diferencia racket de otros lenguajes basados en lisp es la habilidad de extender el lenguaje, es decir

la capacidad de construir nuevos lenguajes ya sean de uso global o especifico del dominio. Esto permite la creacion de un modulo llamado PLT Redex, un lenguaje de uso especifico para especificar y debuggear semanticas operacionales. Con tan solo escribir la gramatica y las reglas de recuccion, y redex te permite probar interactivamente los terminos y generar test aleatorios para intentar generar contraejemplos.

Durante este trabajo usamos estas cualidades de racket para formalizar y intentar falsificar propiedades como progreso. Tanto de manera automatica como con nuestro propio suite de test basado en el que usan en la especificacion de Crystal.

Crystal-Clear y mecanización

3 Crystal-Clear

3.1. Especificación

P ::= *P P P ...*
| *e*
| *s*

e ::= *v* | *var* | *isa? t P*
| *var = P* | *t var = P*
| *unop P* | *P binop P*

s ::= *while P P*
| *if P then P else P*

var ::= *Name*

v ::= *nil* | *bool* | *int32* | *str*

t ::= *Nil* | *Bool* | *Int32* | *String* | *Union* | *Unit*

st ::= *Nil* | *Bool* | *Int32* | *String*

union ::= *st st st ...*

bool ::= *true* | *false*

int32 ::= *integer*

str ::= *string*

binop ::= *shortbinop* | *strictbinop*

strictbinop ::= + | - | * | /
| < | <= | > | >= | ==

shortbinop ::= *and* | *or*

unop ::= - | *not*

r ::= *(ref natural)*

rp ::= *(r v)*

rn ::= *Name r*

ϵ ::= *((Name r) ...)*

3.2. Seguridad

En esta sección intentaremos demostrar que Crystal-Clear cumple con la propiedad seguridad.

Definition 3.2.1 *SEGURIDAD: para cualquier termino bien tipado del lenguaje no es posible llegar a un estado atascado siguiendo las reglas de reducción. Esto significa que el programa no es un valor final pero tampoco se pueda aplicar las reglas de reducción definidas.*

Para ello definiremos dos propiedades.

Definition 3.2.2 *PROGRESO: para cualquier termino si esta bien tipado entonces el termino no es atascado.*

Definition 3.2.3 *PRESERVACIÓN: para cualquier termino si este esta bien tipado y sigue un paso de las reglas de reducción entonces el resultado esta bien tipado.*

Comencemos demostrando progreso. Primero definamos que significa el predicado con nuestros objetos matemáticos.

Theorem 3.2.1 (Progreso) *supongamos que P* $\Gamma \vdash P : t : \Gamma'$ *entonces P es un valor o para algún* σ *y* ϵ *tal que* $\Gamma \vdash \sigma, \epsilon$ *existe una única configuración* $\sigma' : \epsilon' : P'$.

$\sigma : \epsilon : P \rightarrow^{Full} \sigma' : \epsilon' : P'$.

Demostración. Esto se puede probar haciendo inducción en el árbol de la prueba de $\Gamma \vdash P : t : \Gamma'$. Para T-NIL, T-BOOL, T-INT32 y T-STRING son valores por lo tanto ya quedan demostrados que cumplen.

T-NAME: En principio es fácil observar que si P es de la forma *Name* y $\Gamma \vdash$ *Name* $: t :$. Entonces *Name* $\in dom(\Gamma)$. Como $\Gamma \vdash \sigma, \epsilon$ por lo tanto *Name* $\in Dom(\epsilon)$ y su referencia *ref* $\in Dom(\sigma)$. Por lo tanto:

$\sigma : \epsilon :$ *Name* $\rightarrow^{\sigma\epsilon} \sigma : \epsilon :$ v por Dename.

Donde v es el valor relacionado con *Name*. Por otro lado P = E ⟦ *Name* ⟧ con E = ⟦ ⟧. Entonces aplica:

$\sigma : \epsilon :$ [*Name*] $\rightarrow^{Full} \sigma : \epsilon :$ [v] por $FWD - \sigma$.

por lemma 3.2.3 sabemos que esta es la única forma de descomponer este P y, dado que demostramos que se pueden aplicar reglas de $\sigma - progs$ solo se usan reglas de esta relación. restaría demostrar que no hay otra regla aplicable de $\sigma - progs$. Esto se cumple ya que solo una regla es igual a este P.

Sea P igual *Name* = P′ entonces cuando decimos $\Gamma \vdash$ *Name* $= P' : t : \Gamma''$ pueden pasar 2 cosas:

1. Γ contiene *Name* por lo tanto la regla que se aplica es T-Define.

2. Γ no contiene *Name* por lo tanto la regla que se aplica es T-Redefine.

Veamos que en ambos casos se cumple que solo hay un único paso de reducción.

Por hipótesis inductiva P′ es un valor o se puede aplicar alguna reducción sobre P′. Si P′ es un valor v y $\Gamma \vdash \sigma, \epsilon$ donde Γ no contiene a *Name* entonces solo podemos aplicar la regla de Naming y nos queda:

$\sigma : \epsilon :$ *Name* = v $\rightarrow^{Full} \sigma' : \epsilon' :$ nil

donde σ' contiene una nueva referencia con valor v y ϵ' contiene a Name con esta nueva referencia. Si *Name* se encontraba dentro de Γ entonces σ' tendría el nuevo valor asociado a la referencia que ya existía en ϵ mientras que este quedaría igual.

Si P′ no es un valor entonces se puede aplicar una regla de reducción tal que.

$\sigma : \epsilon :$ *Name* = ⟦ P′ ⟧ $\rightarrow^{Full} \sigma'' : \epsilon'' :$ *Name* = ⟦ P″ ⟧ ya sea por $FWD - \sigma$ o FWD-pure.

En esta ocasión conseguimos 2 reducciones para el mismo P, pero es fácil darse cuenta que las configuraciones $\sigma : \epsilon :$ P tienen diferentes $\sigma y \epsilon$

T-CONCAT: Si P = P_1 P_2 ... P_N es fácil ver que si $\Gamma \vdash P_1 P_2 ... P_N : t : e$ntonces $\Gamma \vdash P_i : t_i : p$ara todo i entre 1 y n, t_n = t. Por hipótesis inductiva P_1 es un valor o existe una regla de reducción tal que.

$\sigma : \epsilon : P_1 \rightarrow^{Full} \sigma' : \epsilon' : P_1'$

Por lo tanto si P_1 es un valor entonces P quedaría P_2 ... P_N debido a

$\sigma : \epsilon : P_1\ P_2 ...\ P_N \rightarrow^{prog} \sigma' : \epsilon' : P_2 ...\ P_N$ debido a more-e

o P_1 no es un valor entonces P se puede descomponer en P = P_2 ... P_N ⟦ P_1 ⟧. Entonces :

$\sigma : \epsilon : P_2 ...\ P_N$ ⟦ P_1 ⟧ $\rightarrow^{Full} \sigma' : \epsilon' : P_1'\ P_2 ...\ P_N$

T-ARITHOP Si P = P_1 *Arithop* P_2 Entonces por HI solo tenemos dos casos

P_1 es un valor de tipo Int32 por lo que se puede descomponer en P = v *Arithop* ⟦ P_2 ⟧. Entonces

$\sigma : \epsilon :$ v *Arithop* ⟦ P_2 ⟧ $\rightarrow^{Full} \sigma' : \epsilon' :$ v *Arithop* P_2'

P_1 no es un valor por lo tanto P = *Arithop*P_2 ⟦ P_1 ⟧. Entonces

$\sigma : \epsilon :$ *Arithop* P_2 ⟦ P_1 ⟧ $\rightarrow^{Full} \sigma' : \epsilon' : P_1'$ *Arithop* P_2

T-RELOP y T-SHORTBINOP son análogos a T-ARITHOP.

T-NOT Si P = *not* P_1 sabemos que $\Gamma \vdash P : Bool : y$ por HI sabemos que P_1 es un valor booleano o existe una única regla de reducción tal que.

$\sigma : \epsilon : P_1 \rightarrow^{Full} \sigma' : \epsilon' : P_1'$

y P′ = *not* P_1'

T-NEGATIVE es parecido a T-NOT pero $\Gamma \vdash P_1 : Int32 : .$

T-ISA? si P = **isa?** t P_1 entonces $\Gamma \vdash$ **isa?**$tP_1 : Bool : y$ por hipótesis inductiva P_1 es un valor o existe una única regla de reducción. Si P_1 es un valor entonces se aplica la regla de isa? devolviendo un BOOL. Si P_1 no es un valor se aplica

$\sigma : \epsilon :$ **isa?** t $[\![P_1]\!] \rightarrow^{Full} \sigma' : \epsilon' :$ **isa?** t P_1'

y P′ = **isa?** t P_1'.

T-IF si P = **if** P_1 **then** P_2 **else** P_3 entonces $\Gamma \vdash$ **if**P_1**then**P_2**else**$P_3 : t : .$ Por HI tenemos que P_1 tiene 2 opciones:

$\sigma : \epsilon :$ **if**$[\![P_1]\!]$ **then** P_2 **else** $P_3 \rightarrow^{Full} \sigma' : \epsilon' :$ **if** $[\![P_1']\!]$ **then** P_2 **else** P_3

$\sigma : \epsilon :$ **if**$[\![v]\!]$ **then** P_2 **else** $P_3 \rightarrow^{Full} \sigma' : \epsilon' :$ P′

Donde P′ es un $P_1 o P_2$ dependiendo del valor de v.

T-WHILE si P = **while** P_1 P_2 entonces $\Gamma \vdash$ **while**P_1 $P_2 : t : .$ Por HI tenemos que P_1 tiene 1 opción:

$\sigma : \epsilon :$ **while** $[\![P_1]\!]$ $P_2 \rightarrow^{Full} \sigma' : \epsilon' :$ **if** P_1 **then** (P_2 (**while** P_1 P_2)) **else** nil)

□

Theorem 3.2.2 (Preservacion) *si*

$\Gamma \vdash P : t : \Gamma'$

$\Gamma \vdash \sigma, \epsilon$

$\sigma : \epsilon : P \rightarrow \sigma' : \epsilon' : P'$

entonces, para algún $Dom(\Gamma'') \supseteq Dom(\Gamma)$

$\Gamma'' \vdash P' : t : \Gamma'''$

$\Gamma'' \vdash \sigma', \epsilon'$

Demostración. Esto se puede probar haciendo inducción en todos los posibles programas de nuestra gramática. Para T-NIL, T-BOOL, T-INT32 y T-STRING son valores por lo tanto no hay un P′ del cual preocuparnos.

T-NAME: En principio es fácil observar que P es de la forma *Name* y $\Gamma \vdash$ *Name* $: t : \Gamma'$ con $\Gamma \vdash \sigma, \epsilon$. Por Progreso yo se que hay una única regla que aplica a este caso llamada Dename.

$\sigma : \epsilon :$ *Name* $\rightarrow \sigma' : \epsilon' : v$

Sabemos que esta regla no altera σ ni ϵ. Por lo tanto puedo tomar Γ' σ' ϵ' como Γ σ ϵ respectivamente entonces se cumple que $\Gamma' \supseteq \Gamma$ y que $\Gamma' \vdash v : t : \Gamma''$ ya que $\Gamma' \vdash \sigma', \epsilon'$

T-DEFINE/T-REDEFINE: Sea la forma de P igual a *Name* = P_1. Si P_1 no es un valor entonces se cumple por HI que podemos aplicar una regla tal que:

$\sigma : \epsilon : \textit{Name} = P_1 \rightarrow \sigma' : \epsilon' : \textit{Name} = P_1'$

Con σ', ϵ' tal que $\Gamma' \vdash \sigma', \epsilon'$ para algún $\Gamma' \supseteq \Gamma$ y que P_1' cumple $\Gamma' \vdash P_1' : t : \Gamma''$. Estos $\Gamma', \sigma', \epsilon'$ Cumplen también en el caso de P′ como *Name* = P_1'. En el caso de que P_1 es un valor entonces hay 2 posibles reglas que se aplican: DEFINE es el caso en que *Name* no este en Γ, por lo tanto, lo agrega creando Γ′ esto cumple con $\Gamma' \supseteq \Gamma$. REDEFINE es el caso en el *Name* este en Γ, por lo que cambia el valor, pero que solo altera a Γ′ si el tipo valor de P_1 es distinto al tipo que ya tenia definido en Γ. En los dos casos $\sigma'\epsilon'$ tendrán estos cambios ya sea agregando la variable y su valor o cambiando el valor de la variable ya asignada.

T-CONCAT: P = $P_1\ P_2$ ... P_n. En el caso de la concatenación la única regla semantica o computo que se puede aplicar solo se fija en P_1. Hay dos casos

P_1 es un valor y $\Gamma \vdash P_1P_2 : t : \Gamma'$ en cuyo caso:

$\sigma : \epsilon : P_1\ P_2 \rightarrow \sigma : \epsilon : P_2$

Podemos ver que al no cambiar los $\sigma\ \epsilon$ no hay necesidad de que Γ cambie, por lo que $\Gamma = \Gamma_1$. De esto sigue

$$\frac{\Gamma_1 \vdash P_1 : t_1 : \Gamma_2 \qquad \Gamma_2 \vdash P_2 : t_2 : \Gamma_3}{\Gamma \vdash P_1P_2 : t' : \Gamma} \qquad\qquad \frac{\Gamma_1 \vdash P_2 : t_2 : \Gamma_2}{\Gamma'' \vdash P_2 : t' : \Gamma'''}$$

P_1 no es un valor en cuyo caso:

$\sigma : \epsilon : P_1\ P_2 ... P_n \rightarrow \sigma' : \epsilon' : P_1'\ P_2 ... P_n$

Sabemos por HI que $\sigma'\ \epsilon'$ cumplen con $\Gamma' \vdash \sigma', \epsilon'$ y este Γ′ con

$\Gamma' \supseteq \Gamma$ y $\Gamma' \vdash P' : t : c$on P′ = $P_1'\ P_2 ... P_n$

T-ARITHOP sea el caso de que P = P_1 *Arithop* P_2. Análogo a otras pruebas tenemos dos casos. El caso en el que P_1 es un valor v entonces se intenta usar reglas para reducir P_2, a su vez tenemos otros dos casos. Si P_2 es un valor, se ejecuta la operación pero σ, ϵ, Γ no reciben cambios asi que se puede usar la misma configuración. Si P_2 no es un valor entonces :

$\sigma : \epsilon : v\ \textit{Arithop}\ P_2 \rightarrow \sigma' : \epsilon' : v\textit{Arithop}P_2'$

Por HI sabemos que $\sigma'\ \epsilon'$ tiene un Γ′ $talque$ $\Gamma' \vdash \sigma', \epsilon', \Gamma' \vdash P_2' : t : y$ $\Gamma' \supseteq \Gamma$ por lo tanto podemos usar este Γ′ Para P′

El caso en el que P_1 no es un valor es similar al caso de que P_2 no es un valor.

T-RELOP y T-SHORTBINOP son análogos a T-ARITHOP. La única diferencia en el caso de SHORTBINOP, es que puede no ser necesario saber si P_2 es valor o no.

T-NOT sea P = *Not* P_1. Parecido a otros casos si P_1 es un valor σ, ϵ, Γ no cambian. En el caso de que P_1 no sea un valor entonces por HI tenemos un paso de ejecución único que nos devuelve nuevos $\sigma', \epsilon', \Gamma'$ y utilizamos esos.

T-NEGATIVE es parecido a T-NOT.

T-ISA? Para P = **isa?** t P_1. De nuevo si P_1 es un valor entonces la configuraciones no cambian y podemos reutilizar el mismo Γ. En el caso de que P_1 no sea un valor, se ejecuta un paso y eso nos entrega nuevos $\sigma', \epsilon', \Gamma'$.

T-IF en el caso que P = **if** P_1 **then** P_2 **else** P_3. Como $\Gamma \vdash P : t : \Gamma'$. Entonces existen Γ_1 *y* Γ_2 tal que $\Gamma_1 \vdash P_2 : t : \Gamma_1'$. Para este caso debemos tener en cuenta la relación de tipado definida en IF. Esta relación nos permite hacer reducciones de tipos a partir de P_1, por lemma sabemos que los Γ_1 *y* Γ_2 están contenidos en Γ y por HI.

$\Gamma_1 \vdash \sigma_1, \epsilon_1$

$\Gamma_2 \vdash \sigma_2, \epsilon_2$

Entonces

$\sigma_1 : \epsilon 1 : P_2 \rightarrow \sigma_1' : \epsilon_1' : P_2$

$\sigma_2 : \epsilon 2 : P_3 \rightarrow \sigma_2' : \epsilon_2' : P_3$

T-WHILE a

□

Queremos presentar de un lemma que habla sobre propiedades de términos que emanan de su estructura pero sin asumir si están bien tipado. Como no asumimos que el termino esta bien tipado no podemos asumir si es o no un redex. De todos modos si puede suceder que un termino encaje con el lado izquierdo pero que no sea un redex: es decir que no satisfaga las condiciones laterales que se tienen que satisfacer para que se pueda aplicar una regla semántica. Nosotros queremos hablar solo de si un termino encaja con el lado izquierda o no. independientemente de el cumplimiento de las condiciones laterales. (por ejemplo caso True + 1)

Para demostrar el lemma de descomposición único deberemos definir alguna forma de hablar sobre los redex que encontramos en el libro de PLT-redex.

Pero en nuestro caso necesitamos hablar de un concepto mas general que redex ya que no sabemos si cumplimos con condiciones laterales. Por lo que crearemos un concepto de patrones mas amplio

Definition 3.2.4 *Diremos que* $P \in$ ***Pat*** *si para cierto programa P y cierto patron* ***Pat****.* ***Pat*** *es el lado izquierdo de una de las reglas de la relación progs y P encaja con* ***Pat*** *o para algún patron* $\sigma : \epsilon :$ ***Pat*** *del lado izquierdo de una regla de la relación* $\sigma - progs\ P$ *encaja con ese patron* ***Pat****.*

Esto nos permite definir el siguiente Lemma.

Lemma 3.2.3 (Unico contexto de evaluacion) *Todo programa P es un valor, o existe un único contexto de evaluación E talque P = E ⟦ P′ ⟧ donde P′ ∈* ***Pat****.*

Demostración. Se puede probar haciendo inducción en la estructura de los programas. Los casos bases P = int32 | bool | nil | string ya son valores y no hace falta demostrar nada.

En caso de que P = *Name* solo se puede descomponer en P = ⟦ *Name* ⟧ por definición de E. Por regla DENAME esta en $\sigma - progs$

En el caso de P igual a *Name* = P′. Por HI P′ es un valor o se puede descomponer en P′ = E ⟦ P″ ⟧ donde P'' ∈ **Pat**. Si P′ es un valor v entonces P = ⟦ *Name* = v ⟧, *Name* = v ∈ **Pat** para alguna configuración $\sigma : \epsilon :$ **Pat**. Por reglas REASSIGN y ASSIGN de $\sigma - progs$

En el caso P igual a $P_1\ P_2$ ... P_n. Tenemos multiples formas de descomponer pero la única que se encuentra en E es $[\![P_1]\!]\ P_2 ... P_n$. Por HI P_1 es un valor o existe un único contexto de evaluación E tal que cumpla con estas propiedades. Por lo tanto queda $[\![P_1]\!]\ E' P_2 ... P_n$. En caso de que sea un valor v nos queda $[\![v]\!]\ P_2 ... P_n$ y se aplica la regla de progs more-e.

En el caso P igual P_1 *Binop* P_2 tenemos dos posibles descomposiciones.

P = $[\![P_1]\!]$ *Binop* P_2

P = *v Binop* $[\![P_2]\!]$ En caso que P_1 sea *v*.

Notemos que en ambos caso por Hi se cumple que P_1 es un valor o encaja en el lado izquierdo de una de las reglas progs o $\sigma - progs$.

En el caso de que P sea P_1 *Shortbinop* P_2 es analógico al caso de *Binop*.

Para P = *Not* P_1 descomponemos solo tenemos una forma de descomponer P = *Not* ⟦ P_1 ⟧.

Para el caso de P = - P_1 es análogo al caso anterior.

Cuando P es **isa?** t P_1 solo se puede descomponer de la forma P = **isa?** t ⟦ P_1 ⟧.

En el caso de que P sea **if** P_1 **then** P_2 **else** P_3 solo tiene una forma de descomponer y esa es:

P = **if** $[\![P_1]\!]$ **then** P_2 **else** P_3

En el caso de que P = **while** P_1 P_2 es análogo al caso del **if**.

□

Trabajo relacionado y conclusiones

Class Options, Commands and Environments

4 Class Options

In this chapter I will describe the most common options used, both the ones inherited from `scrbook` and the `kao`-specific ones. Options passed to the class modifies its default behaviour; beware though that some options may lead to unexpected results...

4.1. KOMA Options

The `kaobook` class is based on `scrbook`, therefore it understands all of the options you would normally pass to that class. If you have a lot of patience, you can read the KOMA-Script guide.[1] Actually, the reading of such guide is suggested as it is very instructive.

1: The guide can be downloaded from https://ctan.org/pkg/koma-script?lang=en.

Every KOMA-Script option you pass to the class when you load it is automatically activated. In addition, in `kaobook` some options have modified default values. For instance, the font size is 9.5pt and the paragraphs are separated by space,[2] not marked by indentation.

2: To be precise, they are sepa ted by half a line worth of spa the `parskip` value is «half».

4.2. kao Options

In the future I plan to add more options to set the paragraph formatting (justified or ragged) and the position of the margins (inner or outer in twoside mode, left or right in oneside mode).[3]

3: As of now, paragraphs are justified, formatted with `\singlespacing` (from the `setspace` package) and `\frenchspacing`.

I take this opportunity to renew the call for help: everyone is encouraged to add features or reimplement existing ones, and to send me the results. You can find the GitHub repository at `https://github.com/fmarotta/kaobook`.

To Do

Implement the `justified` and `margin` options. To be consistent with the KOMA-Script style, they should accept a simple switch as a parameter, where the simple switch should be `true` or `false`, or one of the other standard values for simple switches supported by KOMA-Script. See the KOMA-Script documentation for further information.

The above box is an example of a `kaobox`, which will be discussed more thoroughly in Capítulo 9 (Mathematics and Boxes) on page 39. Throughout the book I shall use these boxes to remarks what still needs to be done.

4.3. Other Things Worth Knowing

A bunch of packages are already loaded in the class because they are needed for the implementation. These include:

- etoolbox
- calc
- xifthen
- xkeyval
- xparse
- xstring

Many more packages are loaded, but they will be discussed in due time. Here, we will mention only one more set of packages, needed to change the paragraph formatting (recall that in the future there will be options to change this). In particular, the packages we load are:

- ragged2e
- setspace
- hyphenat
- microtype
- needspace
- xspace
- xcolor (with options `usenames,dvipsnames`)

Some of the above packages do not concern paragraph formatting, but we nevertheless grouped them with the others. By default, the main text is justified and formatted with singlespacing and frenchspacing; the margin text is the same, except that the font is a bit smaller.

As a last warning, please be aware that the `cleveref` package is not compatible with `kaobook`. You should use the commands discussed in Section 7.3 instead.

4.4. Document Structure

We provide optional arguments to the `\title` and `\author` commands so that you can insert short, plain text versions of this fields, which can be used, typically in the half-title or somewhere else in the front matter, through the commands `\@plaintitle` and `\@plainauthor`, respectively. The PDF properties `pdftitle` and `pdfauthor` are automatically set by hyperref to the plain values if present, otherwise to the normal values.[4]

There are defined two page layouts, `margin` and `wide`, and two page styles, `plain` and `fancy`. The layout basically concern the width of the margins, while the style refers to headers and footer; these issues will be discussed in Capítulo 8 (Page Design) on page 34.[5]

The commands `\frontmatter`, `\mainmatter`, and `\backmatter` have been redefined in order to automatically change page layout and style for these sections of the book. The front matter uses the `margin` layout and the `plain` page style. In the mainmatter the margins are wide and the headings are fancy. In the appendix the style and the layout do not change; however we use `\bookmarksetup{startatroot}` so that the bookmarks of the chapters are on the root level (without this, they would be under the preceding part). In the backmatter the margins shrink again and we also reset the bookmarks root.

4: We think that this is an i
tant point so we remark it h
you compile the document
pdflatex, the PDF metadat
be altered so that they mat
plain title and author you
specified; if you did not s
them, the metadata will be
the normal title and author

5: Fo
page
ve w
wide
sent.
and
in fa

5 Margin Stuff

Sidenotes are a distinctive feature of all 1.5-column-layout books. Indeed, having wide margins means that some material can be displayed there. We use margins for all kind of stuff: sidenotes, marginnotes, small tables of contents, citations, and, why not?, special boxes and environments.

5.1. Sidenotes

Sidenotes are like footnotes, except that they go in the margin, where they are more readable. To insert a sidenote, just use the command `\sidenote{Text of the note}`. You can specify a mark[O] with `\sidenote[mark]{Text}`, but you can also specify an offset, which moves the sidenote upwards or downwards, so that the full syntax is:

O: This sidenote has a special mark, a big O!

```
\sidenote[mark][offset]{Text}
```

If you use an offset, you always have to add the brackets for the mark, but they can be empty.[1]

1: If you want to know more about the usage of the `\sidenote` command, read the documentation of the `sidenotes` package.

In `kaobook` we copied a feature from the `snotez` package: the possibility to specify a multiple of `\baselineskip` as an offset. For example, if you want to enter a sidenote with the normal mark and move it upwards one line, type:

```
\sidenote[][*-1]{Text of the sidenote.}
```

As we said, sidenotes are handled through the `sidenotes` package, which in turn relies on the `marginnote` package.

5.2. Marginnotes

This command is very similar to the previous one. You can create a marginnote with `\marginnote[offset]{Text}`, where the offset argument can be left out, or it can be a multiple of `\baselineskip`, *e.g.*

```
\marginnote[-12pt]{Text} or \marginnote[*-3]{Text}
```

To Do

A small thing that needs to be done is to renew the `\sidenote` command so that it takes only one optional argument, the offset. The special mark argument can go somewhere else. In other words, we want the syntax of `\sidenote` to resemble that of `\marginnote`.

We load the packages `marginnote`, `marginfix` and `placeins`. Since `sidenotes` uses `marginnote`, what we said for marginnotes is also valid for sidenotes. Side- and margin- notes are shifted slightly upwards (`\renewcommand{\marginnotevadjust}{3pt}`) in order to allineate them to the bottom of the line of text where the note is issued.

5.3. Footnotes

Even though they are not displayed in the margin, we will discuss about footnotes here, since sidenotes are mainly intended to be a replacement of them. Footnotes force the reader to constantly move from one area of the page to the other. Arguably, marginnotes solve this issue, so you should not use footnotes. Nevertheless, for completeness, we have left the standard command `\footnote`, just in case you want to put a footnote once in a while.*

5.4. Margintoc

Since we are talking about margins, we introduce here the `\margintoc` command, which allows one to put small table

* And this is how they look like. Notice that in the PDF file there is a back reference to the text; pretty cool, uh?

of contents in the margin. Like other commands we have discussed, `\margintoc` accepts a parameter for the vertical offset, like so: `\margintoc[offset]`.

The command can be used in any point of the document, but we think it makes sense to use it just at the beginning of chapters or parts. In this document I make use of a KOMA-Script feature and put it in the chapter preamble, with the following code:

```
\setchapterpreamble[u]{\margintoc}
\chapter{Chapter title}
```

The font used in the margintoc is the same as the one for the chapter entries in the main table of contents at the beginning of the document.

5.5. Marginlisting

On some occasions it may happen that you have a very short piece of code that doesn't look good in the body of the text because it breaks the flow of narration: for that occasions, you can use a `marginlisting`. The support for this feature is still limited, especially for the captions, but you can try the following code:

```
print("Hello World!")
```

```
\begin{marginlisting}[-0.5cm]
\caption{My caption}
\vspace{0.2cm}
\begin{lstlisting}[language=Python,style=kaolstplain]
... code ...
\end{lstlisting}
\end{marginlisting}
```

Unfortunately, the space between the caption and the listing must be adjusted manually; if you find a better way, please let me know.

Not only textual stuff can be displayed in the margin, but also figures. Those will be the focus of the next chapter.

6 Figures and Tables

6.1. Normal Figures and Tables

Figures and tables can be inserted just like in any standard LaTeX document. The `graphicx` package is already loaded and configured in such a way that the figure width is equal to the textwidth and the height is adjusted in order to maintain the original aspect ratio. As you may have imagined, the captions will be positioned. . . well, in the margins. This is achieved with the help of the `floatrow` package.

Here is a picture of Mona Lisa (Figura 6.1), as an example. The captions are formatted as the margin- and the side-notes; If you want to change something about captions you can use the command `\captsetup` from the `caption` package. Remember that if you want to reference a figure, the label must come *after* the caption!

While the format of the caption is managed by `caption`, its position is handled by the `floatrow` package. Achieving this result has been quite hard, but now I am pretty satisfied. In two-side mode, the captions are printed in the correct margin.

Tables can be inserted just as easily as figures, as exemplified by the following code:

The credits for the image above the chapter title go to: Bushra Feroz — Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=68724647

```
\begin{table}
\begin{tabular}{ c c c c }
    \toprule
    col1 & col2 & col3 & col 4 \\
    \midrule
    \multirow{3}{4em}{Multiple row} & cell2 & cell3 & cell4
      \\ &
    cell5 & cell6 & cell7 \\ &
    cell8 & cell9 & cell10 \\
    \multirow{3}{4em}{Multiple row} & cell2 & cell3 & cell4
      \\ &
    cell5 & cell6 & cell7 \\ &
    cell8 & cell9 & cell10 \\
    \bottomrule
\end{tabular}
\end{table}
```

which results in the useless Cuadro 6.1.

Figure 6.1: Caption of a listing.

Cuadro 6.1: A useless table.

col1	col2	col3	col 4
Multiple row	cell2	cell3	cell4
	cell5	cell6	cell7
	cell8	cell9	cell10
Multiple row	cell2	cell3	cell4
	cell5	cell6	cell7
	cell8	cell9	cell10

I don't have much else to say, so I will just insert some blind text. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Figura 6.1: It's Mona Lisa again. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

6.2. Margin Figures and Tables

Marginfigures can be inserted with the environment `marginfigure`. In this case, the whole picture is confined to the margin and the caption is below it. Figura **??** is obtained with something like this:

```
\begin{marginfigure}
    \includegraphics{monalisa}
    \caption[The Mona Lisa]{The Mona Lisa.}
    \ labfig {marginmonalisa}
\end{marginfigure}
```

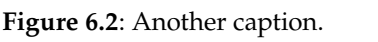


Figure 6.2: Another caption.

There is also the `margintable` environment, of which Cuadro 6.2 is an example. Notice how you can place the caption above the table by just placing the `\caption` command before beginning the `tabular` environment. Usually, figure captions are below, while table captions are above. This rule is also respected for normal figures and tables: the captions are always on the side, but for figure they are aligned to the bottom, while for tables to the top.

Marginfigures and tables can be positioned with an optional offset command, like so:

```
\begin{marginfigure}[ offset ]
    \includegraphics{seaside}
\end{marginfigure}
```

Cuadro 6.2: Another useless table.

col1	col2	col3
Multiple row	cell2	cell3
	cell5	cell6
	cell8	cell9

Offset ca be either a measure or a multiple of `\baselineskip`, much like with `\sidenote`, `\marginnote` and `\margintoc`. If you are wondering how I inserted this orange bubble, have a look at the `todo` package.

Improve this part.

6.3. Wide Figures and Tables

With the environments `figure*` and `table*` you can insert figures which span the whole page width. The caption will be positioned below or above, according to taste.

You may have noticed the full width image at the very beginning of this chapter: that, however, is set up in an entirely different way, which you'll read about in Capítulo 8 on page 34. Now it is time to tackle hyperreferences.

Figura 6.2: A wide seaside, and a wide caption. Credits: By Bushra Feroz — Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=68724647

7 References

7.1. Citations

To cite someone [**Visscher2008**, **James2013**] is very simple: just use the `\sidecite` command. It does not have an offset argument yet, but it probably will in the future. This command supports multiple entries, as you can see, and by default it prints the reference on the margin as well as adding it to the bibliography at the end of the document. Note that the citations have nothing to do with the text,[**James2013**] but they are completely random as they only serve the purpose to illustrate the feature.

For this setup I wrote a separate package, `kaobiblio`, which you can find in the `styles` directory and include in your main tex file. This package accepts all the options that you can pass to `biblatex`, and actually it passes them to `biblatex` under the hood. Moreover, it also defines some commands, like `\sidecite`, and environments that can be used within a `kao` book.[1]

1: For this reason you should always use `kaobiblio` instead of `biblatex`, but the syntax and the options are exactly the same.

As you have seen, the `\sidecite` command will print a citation in the margin. However, this command would be useless without a way to customise the format of the citation, so the `kaobook` provides also the `\formatmargincitation` command. By «renewing» that command, you can choose which items will be printed in the margins. The best way to understand how it works is to see the actual definition of this command.

```
\newcommand{\formatmargincitation}[1]{
    \parencite{#1}: \citeauthor*{#1} (\citeyear{#1}), \citetitle{#1}\\
}
```

Thus, the `\formatmargincitation` accepts one parameter, which is the citation key, and prints the parencite followed by a colon, then the author, then the year (in brackets), and finally the title.[**Battle2014**] Now, suppose that you wish the margin citation to display the year and the author, followed by the title, and finally a fixed arbitrary string; you would add to your document:

```
\renewcommand{\formatmargincitation}[1]{
    \citeyear{#1}, \citeauthor*{#1}: \citetitle{#1}; very interesting!\\
}
```

The above code results in citations that look like the following.[**Zou2005**]
Of course, changing the format is most useful when you also change the default bibliography style. For instance, if you want to use the «philosophy-modern» style for your bibliography, you might have something like this in the preamble:

Zou2005, **Zou2005**: **Zou2005**; very interesting!

```
\usepackage[style=philosophy-modern]{styles/kaobiblio}
\renewcommand{\formatmargincitation}[1]{
    \sdcite{#1}\\
}
\addbibresource{main.bib}
```

The commands like `\citeyear`, `\parencite` and `\sdcite` are just examples. A full reference of the available commands can be found in this cheatsheet, under the «Citations» section.

Finally, to compile a document containing citations, you need to use an external tool, which for this class is biber. You need to run the following (assuming that your tex file is called main.tex):

```
$ pdflatex main
$ biber main
$ pdflatex main
```

7.2. Glossaries and Indices

The `kaobook` class loads the packages `glossaries` and `imakeidx`, with which you can add glossaries and indices to your book. For instance, I previously defined some glossary entries and now I am going to use them, like this: computer. `glossaries` also allows you to use acronyms, like the following: this is the full version, Frame per Second (FPS), and this is the short one FPS. These entries will appear in the glossary in the backmatter.

Unless you use Overleaf or some other fancy IDE for LaTeX, you need to run an external command from your terminal in order to compile a document with a glossary. In particular, the commands required are:[2]

2: These are the commands you would run in a UNIX system; I have no idea on how it works in Windows.

```
$ pdflatex main
$ makeglossaries main
$ pdflatex main
```

Note that you need not run `makeglossaries` every time you compile your document, but only when you change the glossary entries.

To create an index, you need to insert the command \index {subject} whenever you are talking about «subject» in the text. For instance, at the start of this paragraph I would write index{index}, and an entry would be added to the Index in the backmatter. Check it out!

A nomenclature is just a special kind of index; you can find one at the end of this book. To insert a nomenclature, we use the package `nomencl` and add the terms with the command `\nomenclature`. We put then a `\printnomenclature` where we want it to appear.

Also with this package we need to run an external command to compile the document, otherwise the nomenclature will not appear:

```
$ pdflatex main
$ makeindex main.nlo -s nomencl.ist -o main.nls
$ pdflatex main
```

These packages are all loaded in packages.sty, one of the files that come with this class. However, the configuration of the elements is best done in the main.tex file, since each book will have different entries and styles.

Note that the `nomencl` package caused problems when the document was compiled, so, to make a long story short, I had to prevent `scrhack` to load the hack-file for `nomencl`. When compiling the document on Overleaf, however, this problem seem to vanish.

In theory, you would need to r
an external command for the
dex as well, but luckily the pac
ge we suggested, `imakeidx`, c
compile the index automatical

This brief section was by no mea
a complete reference on the su
ject, therefore you should cons
the documentation of the abo
package to gain a full underst
ding of how they work.

7.3. Hyperreferences

In this class we provide a handy sub-package to help you referencing the same elements always in the same way, for consistency across the book. First, you can label each element with a specific command. For instance, should you want to label a chapter, you would put \labch{chapter−title} right after

the `\chapter` directive. This is just a convienence, because `\labch` is actually just an alias to `\label{ch:chapter-title}`, so it spares you the writing of «ch». We defined similar commands for many typically labeled elements, including:

- Page: `\labpage`
- Part: `\labpart`
- Chapter: `\labch`
- Section: `\labsec`
- Figure: `\labfig`
- Table: `\labtab`
- Definition: `\labdef`
- Theorem: `\labthm`
- Proposition: `\labprop`
- Lemma: `\lablemma`
- Remark: `\labremark`
- Example: `\labexample`
- Exercise: `\labexercise`

Of course, we have similar commands for referencing those elements. However, since the style of the reference should depend on the context, we provide different commands to reference the same thing. For instance, in some occasions you may want to reference the chapter by name, but other times you want to reference it only by number. In general, there are four reference style, which we call plain, vario, name, and full.

The plain style references only by number. It is accessed, for chapters, with `\refch{chapter-title}` (for other elements, the syntax is analogous). Such a reference results in: Capítulo 7.

The vario and name styles rest upon the `varioref` package. Their syntax is `\vrefch{chapter-title}` and `\nrefch{chapter-title}`, and they result in: Capítulo 7 on page 29, for the vario style, and: Capítulo 7 (References), for the name style. As you can see, the page is referenced in `varioref` style.

The full style references everything. You can use it with `\frefch{chapter-title}` and it looks like this: Capítulo 7 (References) on page 29.

Of course, all the other elements have similar commands (*e.g.* for parts you would use `\vrefpart{part-title}` or something like that). However, not all elements implement all the four styles. The commands provided should be enough, but if you want to see what is available or to add the missing ones, have a look at the attached package.

Design and Additional Features

8 Page Design

8.1. Headings

So far, in this document I used two different styles for the chapter headings: one has the chapter name, a rule and, in the margin, the chapter number; the other has an image at the top of the page, and the chapter title is printed in a box (like for this chapter). There is one additional style, which I used only in the appendix (on page 44); there, the chapter title is enclosed in two horizontal rules, and the chapter number (or letter, in the case of the appendix) is above it.[1]

1: To be honest, I do not think that mixing heading styles like this is a wise choice, but in this document I did it only to show you how they look.

Every book is unique, so it makes sense to have different styles from which to choose. Actually, it would be awesome if whenever a `kao`-user designs a new heading style, he or she added it to the three styles already present, so that it will be available for new users and new books.

The choice of the style is made simple by the `\setchapterstyle` command. It accepts one option, the name of the style, which can be: «plain», «kao», or «lines».[2] If instead you want the image style, you have to use the command `\setchapterimage`, which accepts the path to the image as argument; you can also provide an optional parameter in square brackets to specify the height of the image.

2: Plain is the default LaTeX title style; the other ones are self explanatory.

Let us make some examples. In this book, I begin a normal chapter with the lines:

```
\setchapterstyle{kao}
\setchapterpreamble[u]{\margintoc}
\chapter{Title of the Chapter}
```

```
4 \labch{ title }
```

In Line 1 I choose the style for the title to be «kao». Then, I specify that I want the margin toc. The rest is ordinary administration in LaTeX, except that I use my own `\labch` to label the chapter. Actually, the `\setchapterpreamble` is a standard KOMA-Script one, so I invide you to read about it in the KOMA documentation. Once the chapter style is set, it holds until you change it.[3] Whenever I want to start a chapter with an image, I simply write:

```
1 \setchapterimage[7cm]{path/to/image.png}  % Optionally
      specify the height
2 \setchapterpreamble[u]{\margintoc}
3 \chapter{Catchy Title}  % No need to set a chapter style
4 \labch{catchy}
```

If you prefer, you can also specify the style at the beginning of the main document, and that style will hold until you change it again.

3: The `\margintoc` has to be specified at every chapter. Perhaps in the future this may change; it all depends on how this feature will be welcomed by the users, so keep in touch with me if you have preferences!

8.2. Headers & Footers

Headers and footers in KOMA-Script are handled by the `scrlayer-scrpage` package. There are two basic style: «scrheadings» and «plain.scrheadings». The former is used for normal pages, whereas the latter is used in title pages (those where a new chapter starts, for instance) and, at least in this book, in the front matter. At any rate, the style can be changed with the `\pagestyle` command, *e.g.* \pagestyle{plain.scrheadings}.

In both stles, the footer is completely empty. In plain.scrheadings, also the header is absent (otherwise it wouldn't be so plain. . .), but in the normal style the design is reminescent of the «kao» style for chapter titles.

To Do

The `twoside` class option is still unstable and may lead to unexpected behaviours. As always, any help will be greatly appreciated.

Entry	Command to Activate
Table of Contents	\setuptoc{toc}{totoc}
List of Figs and Tabs	\PassOptionsToClass{toc=listof}{\@baseclass}
Bibliography	\PassOptionsToClass{toc=bibliography}{\@baseclass}

Cuadro 8.1: Commands to add a particular entry to the table of contents.

8.3. Table of Contents

Another important part of a book is the table of contents. By default, in `kaobook` there is an entry for everything: list of figures, list of tables, bibliographies, and even the table of contents itself. Not everybody might like this, so we will provide a description of the changes you need to do in order to enable or disable each of these entries. In the following Cuadro 8.1, each item corresponds to a possible entry in the TOC, and its description is the command you need to provide to have such entry. These commands are specified in the attached style package,[4] so if you don't want the entries, just comment the corresponding lines.

4: In the same file, you can also choose the titles of these entries.

Of course, some packages, like those for glossaries and indices, will try to add their own entries. In such cases, you have to follow the instructions specific to that package. Here, since we have talked about glossaries and notations in Capítulo 7, we will biefly see how to configure them.

In a later section, we will se you can define your own fl environment, and endow i an entry in the TOC.

For the `glossaries` package, use the «toc» option when you load it: \usepackage[toc]{glossaries}. For `nomencl`, pass the «intoc» option at the moment of loading the package. Both `glossaries` and `nomencl` are loaded in the attached «packages» package.

Additional configuration of the table of contents can be performed through the packages `etoc`, which is loaded because it is needed for the margintocs, or the more traditional `tocbase`. Read the respective documentations if you want to be able to change the default TOC style.[5]

8.4. Page Layout

Besides the page style, you can also change the width of the content of a page. This is particularly useful for pages dedicated to part titles, where having the 1.5-column layout might be a little awkward, or for pages where you only put

figures, where it is important to exploit all the available space.

In practice, there are two layouts: «wide» and «margin». The former suppresses the margins and allocates the full page for contents, while the latter is the layout used in most of the pages of this book, including this one. The wide layout is also used automatically in the front and back matters.

To change page layout, use the `\pagelayout` command. For example, when I start a new part, I write:

```
\pagelayout{wide}
\addpart{Title of the New Part}
\pagelayout{margin}
```

8.5. Numbers & Counters

In this short section we shall see how dispositions, sidenotes and figures are numbered in the `kaobook` class.

By default, dispositions are numbered up to the section. This is achieved by setting: \setcounter{secnumdepth}{1}.

The sidenotes counter is the same across all the document, but if you want it to reset at each chapter, just uncomment the line

```
\counterwithin*{sidenote}{chapter}
```

in the `styles/style.sty` package provided by this class.

Figure and Table numbering is also per-chapter; to change that, use something like:

```
\renewcommand{\thefigure}{\arabic{section}.\arabic{
    figure}}
```

8.6. White Space

One of the things that I find most hard in LaTeX is to finely tune the white space around objects. There are not fixed rules, each object needs its own adjustment. Here we shall see how some spaces are defined at the moment in this class.

Attention! This section may be incomplete.

Space around figures and tables

```
\renewcommand\FBaskip{.4\topskip}
\renewcommand\FBbskip{\FBaskip}
```

Space around captions

```
\captionsetup{
    aboveskip=6pt,
    belowskip=6pt
}
```

Space around displays (*e.g.* equations)

```
\setlength\abovedisplayskip{6pt plus 2pt minus 4pt}
\setlength\belowdisplayskip{6pt plus 2pt minus 4pt}
\abovedisplayskip 10\p@ \@plus2\p@ \@minus5\p@
\abovedisplayshortskip \z@ \@plus3\p@
\belowdisplayskip \abovedisplayskip
\belowdisplayshortskip 6\p@ \@plus3\p@ \@minus3\p@
```

Mathematics and Boxes 9

9.1. Theorems

Despite most people complain at the sight of a book full of equations, mathematics is an important part of many books. Here, we shall illustrate some of the possibilities. We believe that theorems, definitions, remarks and examples should be emphasised with a shaded background; however, the colour should not be to heavy on the eyes, so we have chosen a sort of light yellow.[1]

1: The boxes are all of the same colour here, because we did not want our document to look like Harlequin.

Definition 9.1.1 *Let (X, d) be a metric space. A subset $U \subset X$ is an open set if, for any $x \in U$ there exists $r > 0$ such that $B(x, r) \subset U$. We call the topology associated to d the set τ_d of all the open subsets of (X, d).*

Definition 9.1.1 is very important. I am not joking, but I have inserted this phrase only to show how to reference definitions. The following statement is repeated over and over in different environments.

Theorem 9.1.1 *A finite intersection of open sets of (X, d) is an open set of (X, d), i.e τ_d is closed under finite intersections. Any union of open sets of (X, d) is an open set of (X, d).*

Proposition 9.1.2 *A finite intersection of open sets of (X, d) is an open set of (X, d), i.e τ_d is closed under finite intersections. Any union of open sets of (X, d) is an open set of (X, d).*

Lemma 9.1.3 *A finite intersection[a] of open sets of (X, d) is an open set of (X, d), i.e τ_d is closed under finite intersections. Any union of open sets of (X, d) is an open set of (X, d).*

[a] I'm a footnote

You can safely ignore the content of the theorems. . . I assume that if you are interested in having theorems in your book, you already know something about the classical way to add

them. These example should just showcase all the things you can do within this class.

Corollary 9.1.4 (Finite Intersection, Countable Union) *A finite intersection of open sets of (X, d) is an open set of (X, d), i.e τ_d is closed under finite intersections. Any union of open sets of (X, d) is an open set of (X, d).*

Demostración. The proof is left to the reader as a trivial exercise. Hint: Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. □

Definition 9.1.2 *Let (X, d) be a metric space. A subset $U \subset X$ is an open set if, for any $x \in U$ there exists $r > 0$ such that $B(x, r) \subset U$. We call the topology associated to d the set τ_d of all the open subsets of (X, d).*

Example 9.1.1 Let (X, d) be a metric space. A subset $U \subset X$ is an open set if, for any $x \in U$ there exists $r > 0$ such that $B(x, r) \subset U$. We call the topology associated to d the set τ_{d} of all the open subsets of (X, d).

Remark 9.1.1 Let (X, d) be a metric space. A subset $U \subset X$ is an open set if, for any $x \in U$ there exists $r > 0$ such that $B(x, r) \subset U$. We call the topology associated to d the set τ_{d} of all the open subsets of (X, d).

As you may have noticed, definitions, example and remarks have independent counters; theorems, propositions, lemmas and corollaries share the same counter.

Remark 9.1.2 Here is how an integral looks like inline: $\int_a^b x^2 dx$, and here is the same integral displayed in its own paragraph:

$$\int_a^b x^2 dx$$

We provide two files for the theorem styles: plaintheorems.sty, which you should include if you do not want coloured boxes around theorems; and mdftheorems.sty, which is the one used for this document.[2] Of course, you will have to edit these files according to your taste and the general style of the book.

2: The plain one is not showed, but actually it is exactly the same as this one, only without the yellow boxes.

9.2. Boxes & Custom Environments [3]

Say you want to insert a special section, an optional content or just something you want to emphasise. We think that nothing works better than a box in these cases. We used `mdframed` to construct the ones shown below. You can create and modify such environments by editing the provided file environments.sty.

Title of the box

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

If you set up a counter, you can even create your own numbered environment.

Comment 9.2.1

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

9.3. Experiments

It is possible to wrap marginnotes inside boxes, too. Audacious readers are encouraged to try their own experiments and let me know the outcomes.

title of margin note

Margin note inside a kaobox (Actually, kaobox inside a marginnote!)

I believe that many other special things are possible with the `kaobook` class. During its development, I struggled to keep it as flexible as possible, so that new features could be added without too great an effort. Therefore, I hope that you can find the optimal way to express yourselves in writing a book, report or thesis with this class, and I am eager to see the outcomes of any experiment that you may try.

Appendix

A

Heading on level 0 (chapter)

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

A.1. Heading on level 1 (section)

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Heading on level 2 (subsection)

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Heading on level 3 (subsubsection)

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Heading on level 4 (paragraph) Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque

placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

A.2. Lists

Example for list (itemize)

- First itemtext
- Second itemtext
- Last itemtext
- First itemtext
- Second itemtext

Example for list (4*itemize)

- First itemtext
  - First itemtext
    - First itemtext
      - First itemtext
      - Second itemtext
    - Last itemtext
  - First itemtext
- Second itemtext

Example for list (enumerate)

1. First itemtext
2. Second itemtext
3. Last itemtext
4. First itemtext
5. Second itemtext

Example for list (4*enumerate)

1. First itemtext
   a) First itemtext
      1) First itemtext
         a′ First itemtext

b′ Second itemtext

2) Last itemtext

b) First itemtext

2. Second itemtext

Example for list (description)

First itemtext
Second itemtext
Last itemtext
First itemtext
Second itemtext

Example for list (4*description)

First itemtext

First itemtext

First itemtext

First itemtext
Second itemtext

Last itemtext

First itemtext

Second itemtext

Notation

The next list describes several symbols that will be later used within the body of the document.

c Speed of light in a vacuum inertial frame

h Planck constant

Greek Letters with Pronounciation

Character	Name	Character	Name
α	alpha *AL-fuh*	ν	nu *NEW*
β	beta *BAY-tuh*	ξ, Ξ	xi *KSIGH*
γ, Γ	gamma *GAM-muh*	o	omicron *OM-uh-CRON*
δ, Δ	delta *DEL-tuh*	π, Π	pi *PIE*
ϵ	epsilon *EP-suh-lon*	ρ	rho *ROW*
ζ	zeta *ZAY-tuh*	σ, Σ	sigma *SIG-muh*
η	eta *AY-tuh*	τ	tau *TOW (as in cow)*
θ, Θ	theta *THAY-tuh*	υ, Υ	upsilon *OOP-suh-LON*
ι	iota *eye-OH-tuh*	ϕ, Φ	phi *FEE, or FI (as in hi)*
κ	kappa *KAP-uh*	χ	chi *KI (as in hi)*
λ, Λ	lambda *LAM-duh*	ψ, Ψ	psi *SIGH, or PSIGH*
μ	mu *MEW*	ω, Ω	omega *oh-MAY-guh*

Capitals shown are the ones that differ from Roman capitals.

Alphabetical Index