Penrose: From Mathematical Notation to Beautiful Diagrams (Supplemental Material)

KATHERINE YE, Carnegie Mellon University
WODE NI, Carnegie Mellon University
MAX KRIEGER, Carnegie Mellon University
DOR MA'AYAN, Technion and Carnegie Mellon University
JENNA WISE, Carnegie Mellon University
JONATHAN ALDRICH, Carnegie Mellon University
JOSHUA SUNSHINE, Carnegie Mellon University
KEENAN CRANE, Carnegie Mellon University

1 GRAMMAR

This document contains grammars for the three languages that Penrose supports, Domain, Substance, and Style. Notation used:

- The syntax $\overline{x}$ denotes a list of elements x.
- Normal text denotes a set (*e.g.*, TypeVars) or uninterpreted symbol (*e.g.*, Top for the supertype of all types).
- *Italicized*, hollow, or **bold** text denotes a form (*e.g.*, *S*, the form of Domain statements).
- `Monotype` text denotes concrete syntax (*e.g.*, the keyword `predicate`).

1.1 Domain Grammar

The Domain language enables domain experts to parametrize the Substance language by declaring the constructs in a domain. The grammar can be found in Figure 1.

X	$\in$	TypeVars	*(type variables)*
x	$\in$	Vars	*(variables)*
s	$\in$	SubFields	*(Substance fields)*
c, f, p	$\in$	Decls	*(declaration names)*
np	$\in$	NotationPatterns	*(notation patterns)*
T	$::=$	X | Top | Prop	*(types)*
cd	$::=$	`type` T	*(type cons decl stmt)*
std	$::=$	`T <: T`	*(sub type decl stmt)*
vd	$::=$	`constructor` $c : \overline{(s : T)} \to T$	*(value cons decl stmt)*
od	$::=$	`function` $f : \overline{(x : T)} \to T$	*(function decl stmt)*
pd	$::=$	`predicate` $p : \overline{(x : T)}$	*(predicate decl stmts)*
sn	$::=$	`Notation` $np \sim \mathbf{P}_{Substance}$	*(statement notation)*
val	$::=$	`value` x : T	*(value prelude)*
S	$::=$	$cd \mid std \mid vd \mid od \mid pd \mid sn \mid val$	*(statements)*
$\mathbf{P}$	$::=$	$\overline{S}$	*(program)*

Fig. 1. Domain grammar

1.2 Substance Grammar

The Substance language enables novice users to express abstract logical relationships in the language of that domain. Substance programs are parametrized by Domain schemas and are visualized according to Style programs. The grammar of the Substance language can be found in Figure 2. Note that it refers to variables y and types T defined in the Domain grammar.

x	$\in$	Vars	*(variables)*
s	$\in$	SubFields	*(Substance fields)*
lt	$::=$	StringLit	*(literals)*
E	$::=$	$x \mid f(\overline{E}) \mid c(\overline{E})$	*(expressions)*
D	$::=$	$x.s$	*(deconstructors)*
L	$::=$	`NoLabel` x | `AutoLabel` x | `AutoLabel All` | `Label` x lt	*(labels)*
Q	$::=$	$p(\overline{E}) \mid p(\overline{Q})$	*(predicates)*
S	$::=$	$T\ x \mid x := E \mid x := D \mid Q \mid L$	*(statements)*
$\mathbf{P}$	$::=$	$\overline{S}$	*(program)*

Fig. 2. Substance language grammar

1.3 Style Grammar

Style programs define the visual representation of the Substance programs expressible in a Domain schema via a stylesheet-like mechanism. The grammar of Style programs is given in Fig. 3, which refers to the grammars for selectors (Fig. 4) and blocks (Fig. 5).

The grammar of Style selectors is very similar to the combined Domain and Substance grammars, with two differences. First, value variables may either refer directly to a Substance variable x or bind a new Style variable y. Second, Substance type declarations $\mathbb{S}_o$ have been split from other kinds of Substance statements $\mathbb{S}_r$. (We omit the `override` flag from the core language.)

y	$\in$	StyVars	*(Style variables)*
H	$::=$	$y \mid \mathcal{S}$	*(a header is a namespace or a selector)*
$\mathcal{B}$	$::=$	$(H, \mathbb{B})$	*(a pair of a header and a block)*
$\mathbb{P}$	$::=$	$\overline{\mathcal{B}}$	*(a Style program)*

Fig. 3. Style program grammar

Authors' addresses: Katherine Ye, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA, 15213; Wode Ni, Carnegie Mellon University; Max Krieger, Carnegie Mellon University; Dor Ma'ayan, Technion and Carnegie Mellon University, Haifa, Israel; Jenna Wise, Carnegie Mellon University; Jonathan Aldrich, Carnegie Mellon University; Joshua Sunshine, Carnegie Mellon University; Keenan Crane, Carnegie Mellon University.

X	$\in$	TypeVars	*(type variables)*
x	$\in$	SubVars	*(Substance variables)*
y	$\in$	StyVars	*(Style variables)*
B	$::=$	$x \mid y$	*(binding forms)*
$\mathbb{T}$	$::=$	X	*(types)*
$\mathbb{E}$	$::=$	$B \mid f(\overline{\mathbb{E}}) \mid c(\overline{\mathbb{E}})$	*(expressions)*
$\mathbb{Q}$	$::=$	$p(\overline{\mathbb{E}}) \mid p(\overline{\mathbb{Q}})$	*(predicates)*
$\mathbb{S}_o$	$::=$	$\mathbb{T}\,B$	*(Substance object declarations)*
$\mathbb{S}_r$	$::=$	$B := \mathbb{E} \mid \mathbb{Q}$	*(statements about Substance objects)*
$\mathcal{S}$	$::=$	$\overline{\mathbb{S}_o}$ `where` $\overline{\mathbb{S}_r}$	*(selectors)*

Fig. 4. Style selector grammar

n	$\in$	StyFields	*(Style block fields and shape properties)*
$\mathbb{C}$	$\in$	StyLibrary	*(constructors for graphical primitives)*
χ	$\in$	StyLibrary	*(function names)*
Float	$::=$	FloatLit $\mid$ VaryingFloat	*(floats, fixed or varying)*
L	$::=$	*Float* $\mid$ BoolLit $\mid$ StringLit $\mid$ IntLit $\mid \ldots$	*(literals)*
ϵ_c	$::=$	$-\epsilon \mid \epsilon + \epsilon \mid \epsilon * \epsilon \mid \ldots$	*(inline computations)*
π	$::=$	$B.n \mid \pi.n$	*(paths)*
ϵ	$::=$	$L \mid \pi \mid \chi(\overline{\epsilon}) \mid \epsilon_c \mid \overline{\epsilon} \mid \mathbb{C}\,\overline{n = \epsilon}$	*(expressions)*
ψ	$::=$	$\pi = \epsilon \mid$ `delete` π	*(statements)*
$\mathbb{B}$	$::=$	$\overline{\psi}$	*(blocks)*

Fig. 5. Style block grammar