Metamorphic Domain-Specific Languages

Mathieu Acher, Benoit Combemale, Philippe Collet

Gemoc

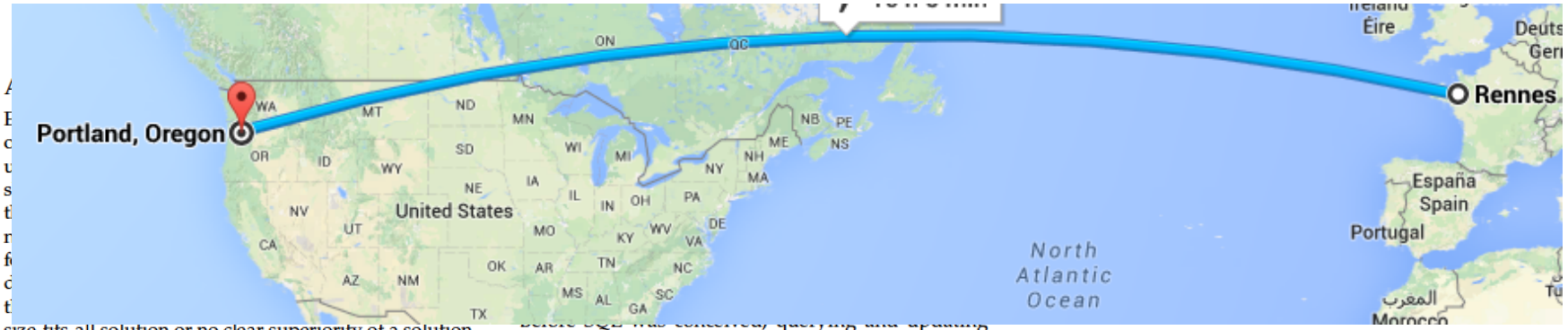
informatiques mathématiques
Inria
UNIVERSITÉ DE RENNES 1
UMR IRISA
Université Nice Sophia Antipolis
i3S sophia antipolis

Metamorphic Domain-Specific Languages

A Journey Into the Shapes of a Language

Mathieu Acher Benoit Combemale
University of Rennes 1, Inria, IRISA, France
mathieu.acher@irisa.fr, benoit.combemale@inria.fr

Philippe Collet
Univ. Nice - Sophia Antipolis, I3S, France
philippe.collet@unice.fr



size-fits-all solution or no clear superiority of a solution compared to another. On the contrary, we found that it does make sense to continue the maintenance of an external and internal DSL. Based on our experience and on an analysis of the DSL engineering field, the vision that we foresee for the future of software languages is their ability to be self-adaptable to the most appropriate shape (including the corresponding integrated development environment) according to a particular usage or task. We call metamorphic DSL such a language, able to change from one shape to another shape.

Categories and Subject Descriptors D. Software [*D.2 SOFTWARE ENGINEERING*]: D.2.6 Programming Environments; D. Software [*D.3 PROGRAMMING LANGUAGES*]: D.3.0 General

Keywords programming; domain-specific languages; metamorphic

1. Introduction

Domain-specific languages (DSLs) are more and more used to leverage business or technical domain exper-

Onward! 2014, October 20–24, 2014, Portland, OR, USA.

http://dx.doi.org/10.1145/2661136.2661159

relational databases with the available programming languages led to a huge semantic gap between data and control processing. With SQL, users can write a query in terms of an implicit algebra without knowing the internal layout of a database. Users can also benefit from performance optimization: a query optimizer can determine the most efficient way to execute a given query.

Another benefit of DSLs is their capacity at improving communication with domain experts [1–3], thus tackling one of the hardest problems in software development. But DSLs are also ordinary languages, in the sense that many difficult design decisions must be taken during their construction and maintenance [1]. They usually can take different shapes: plain old to more fluent Application Programming Interface (APIs) ; internal or embedded DSLs written inside an existing host language ; external DSLs with their own syntax and domain-specific tooling. To keep it simple, a useful and common distinction is to consider that a DSL can come in two main shapes [2]: external or internal. When an API is primarily designed to be readable and to "*flow*", we also consider it as a DSL.

As for SQL – invented in 1974 and one of the first DSLs – it is interesting to note that it comes itself in different shapes. Figure 1 shows three of these shapes on the same basic query example. The top part of the figure shows the plain SQL variant, with a classical *"select, from, where"* clause. In the middle part of the figure, we show the same query written in Java with

Acknowledgements

Thomas Degueule, Guillaume Bécan, Olivier Barais, Julien Richard-Foy, Jean-Marc Jézéquel

DiverSE Diversity-Centric Software Engineering members

and Jonathan Aldrich

Do you know

HTML?

```
object XMLTest1 extends Application {
  val page =
  <html>
    <head>
      <title>Hello XHTML world</title>
    </head>
    <body>
      <h1>Hello world</h1>
      <p><a href="scala-lang.org">Scala</a> talks XHTML</p>
    </body>
  </html>;
  println(page.toString())
}
```

Scala

```
// Import the Glitter DSL
import glitter._

object Templates {

  // Define a reusable layout
  def layout(body: Xml) =
    html5dtd | 'html (
        'head :: 'title :: "Glitter is amazing!"
      | 'body :: body
    )

  // Define a template taking one String argument and using the layout defined above
  def show(name: String) =
    layout (
        'h1 :: "Show user"
      | 'p :: ("Hello " | 'strong(name) | "!")
    )

  // Define a template taking a List of Strings, using the layout defined above
  def index(users: List[String]) =
    layout (
        'h1 :: "User list"
      | 'ul % 'class~"user-list" :: (for (user <- users) yield ('li :: user))
    )
}
```

https://github.com/julienrf/glitter

TCS Wyvern (Omar et al., OOPLSA'14)

```
let webpage : HTML = HTMLElement(Dict.empty(), [BodyElement(Dict.empty(),
  [H1Element(Dict.empty(), [TextNode("Results for " + keyword)]),
  ULElement((Dict.add Dict.empty() ("id","results")), to_list_items(query(db,
    SelectStmt(["title", "snippet"], "products",
      [WhereClause(InPredicate(StringLit(keyword), "title"))]))))])])
```

```
let webpage : HTML = <html><body><h1>Results for {keyword}</h1>
  <ul id="results">{to_list_items(query(db,
    SELECT title, snippet FROM products WHERE {keyword} in title))}
  </ul></body></html>
```

```
let webpage : HTML = parse_html("<html><body><h1>Results for "+keyword+"</h1>
  <ul id=\"results\">" + to_string(to_list_items(query(db, parse_sql(
    "SELECT title, snippet FROM products WHERE '"+keyword+"' in title")))) +
  "</ul></body></html>")
```

```
object XMLTest1 extends Application {
  val page =
  <html>
    <head>
      <title>Hello XHTML world</title>
    </head>
    <body>
      <h1>Hello world</h1>
      <p><a href="scala-lang.org">Scala</a> talks XHTML</p>
    </body>
  </html>;
  println(page.toString())
}
```

```
1  // Import the Glitter DSL
2  import glitter._
3
4  object Templates {
5
6    // Define a reusable layout
7    def layout(body: Xml) =
8      html5dtd | 'html (
9          'head :: 'title :: "Glitter is amazing!"
10       | 'body :: body
11     )
12
13   // Define a template taking one String argument and using the layout defined above
14   def show(name: String) =
15     layout (
16         'h1 :: "Show user"
17       | 'p :: ("Hello " | 'strong(name) | "!")
18     )
19
20   // Define a template taking a list of Strings, using the layout defined above
21   def index(users: List[String]) =
22     layout (
23         'h1 :: "User list"
24       | 'ul % 'class~"user-list" :: (for (user <- users) yield ('li :: user))
25     )
26 }
```

Metamorphic HTML

```
1  let webpage : HTML = <html><body><h1>Results for {keyword}</h1>
2    <ul id="results">{to_list_items(query(db,
3      SELECT title, snippet FROM products WHERE {keyword} in title))}
4    </ul></body></html>
```

Shell | Node.js | Java | Python | PHP | Ruby | Objective-C | Go

```
curl --request POST \
  --url 'http://mockbin.com/request?foo=bar&foo=baz' \
  --header 'accept: application/json' \
  --header 'content-type: application/json' \
  --cookie 'foo=bar; bar=baz' \
  --data '{"foo": "bar"}'
```

Shell | Node.js | Java | Python | PHP | Ruby | Objective-C | Go

```
HttpResponse<String> response = Unirest.post("http://mockbin.com/request?foo=bar&foo=baz")
  .header("cookie", "foo=bar; bar=baz")
  .header("accept", "application/json")
  .header("content-type", "application/json")
  .body("{\"foo\": \"bar\"}")
  .asString();
```

Shell | Node.js | Java | Python | PHP | Ruby | Objective-C | Go

```
#import <Foundation/Foundation.h>

NSDictionary *headers = @{ @"cookie": @"foo=bar; bar=baz",
                           @"accept": @"application/json",
                           @"content-type": @"application/json" };
NSDictionary *parameters = @{ @"foo": @"bar" };

NSData *postData = [NSJSONSerialization dataWithJSONObject:parameters options:0 error:nil];

NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:[NSURL URLWithString:@"ht
                                                    cachePolicy:NSURLRequestUseProtocolCa
                                                timeoutInterval:10.0];
[request setHTTPMethod:@"POST"];
[request setAllHTTPHeaderFields:headers];
[request setHTTPBody:postData];
```

Metamorphic
Mashape

http://blog.mashape.com/apiembed-embeddable-api-code-snippets/

Do you really know

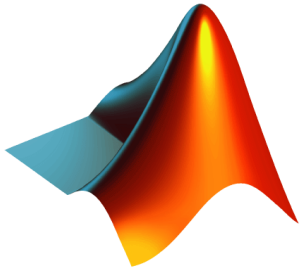
DSLs?

BibTeX

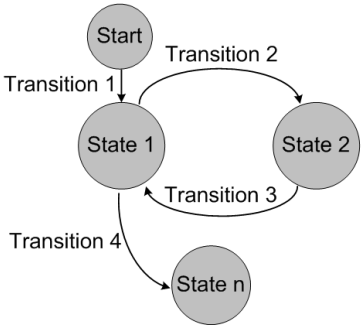
Graphviz

HTML

Make



Matlab



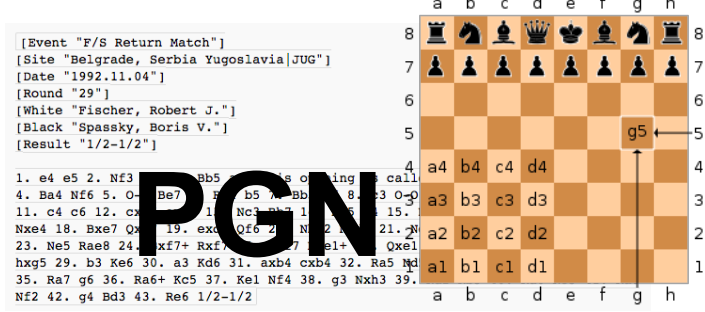
Finite State Machine

SQL

CSS

PGN

```
[Event "F/S Return Match"]
[Site "Belgrade, Serbia Yugoslavia|JUG"]
[Date "1992.11.04"]
[Round "29"]
[White "Fischer, Robert J."]
[Black "Spassky, Boris V."]
[Result "1/2-1/2"]

1. e4 e5 2. Nf3 [illegible] Bb5 [illegible] call
4. Ba4 Nf6 5. O- [illegible] Be7 [illegible] b5 [illegible] 8. c3 O-O
11. c4 c6 12. cx [illegible] Nc3 [illegible] 15.
Nxe4 18. Bxe7 Qx [illegible] 19. exd Qf6 2[illegible] N[illegible]2 [illegible] 21. N[illegible]
23. Ne5 Rae8 24. [illegible]xf7+ Rxf[illegible] [illegible]7 [illegible]el+ [illegible] Qxe1
hxg5 29. b3 Ke6 30. a3 Kd6 31. axb4 cxb4 32. Ra5 Nd[illegible]
35. Ra7 g6 36. Ra6+ Kc5 37. Ke1 Nf4 38. g3 Nxh3 39.
Nf2 42. g4 Bd3 43. Re6 1/2-1/2
```



Domain-Specific Languages (DSLs)

DSL =

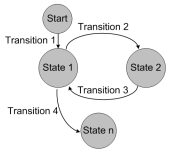
Syntax + Services

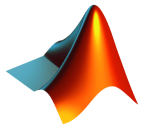
Specialized notation:

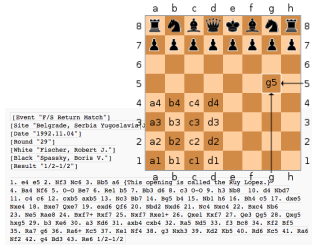
Textual or Graphical
Specific Vocabulary
Idiomatic constructs

Specialized tools/IDE:

Editor with auto-completion, syntax highlighting, etc.
Compiler
Interpreter
Debugger
Profiler
Syntax/Type Checker
…





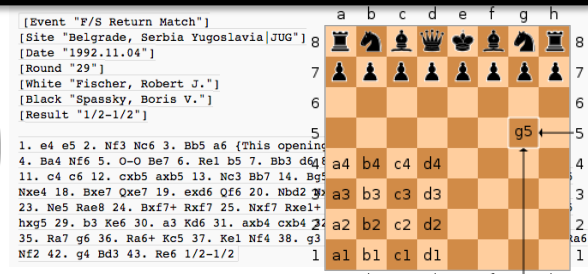


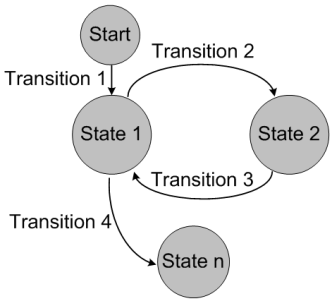
Why DSLs exist?

Because there is no one-size-fits-all solution!

BibTeX

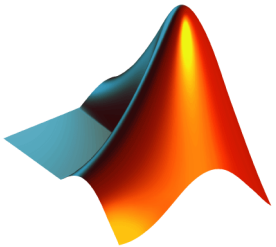
HTML





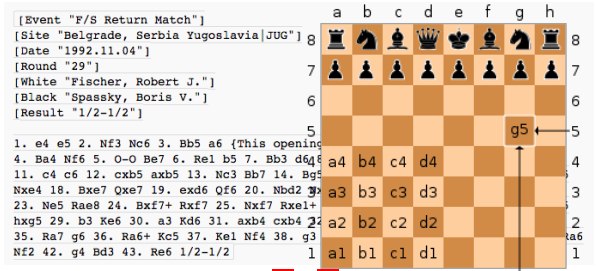
SQL

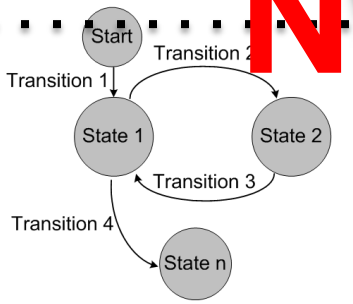
CSS

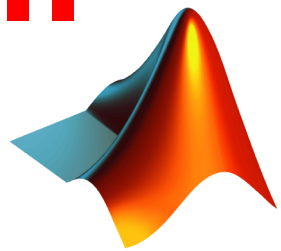


Domain-Specific Languages (DSLs)

General Purpose Languages
(e.g., Java, Scala, Haskell, Ruby)







No one-size-fits-all solution!

We need DSLs, whatever they're

Domain-Specific Languages (DSLs)

Even for a given domain and class of problem, there is no one-size-fits-all solution!

(Java)

(PHP)

(Scala)

(Plain text SQL)

(Clojure)

(C#)

Domain-Specific Language (DSL)

Even for a given domain and class of problem, there is no one-size-fits-all solution!

Polymorphic DSLs: different shapes

(PHP)

(Scala)

(Java)

(Clojure)

(Plain text SQL)

(C#)

```sql
-- Select all books by authors born after 1920,
-- named "Paulo" from a catalogue:
SELECT *
  FROM t_author a
  JOIN t_book b ON a.id = b.author_id
 WHERE a.year_of_birth > 1920
   AND a.first_name = 'Paulo'
 ORDER BY b.title
```

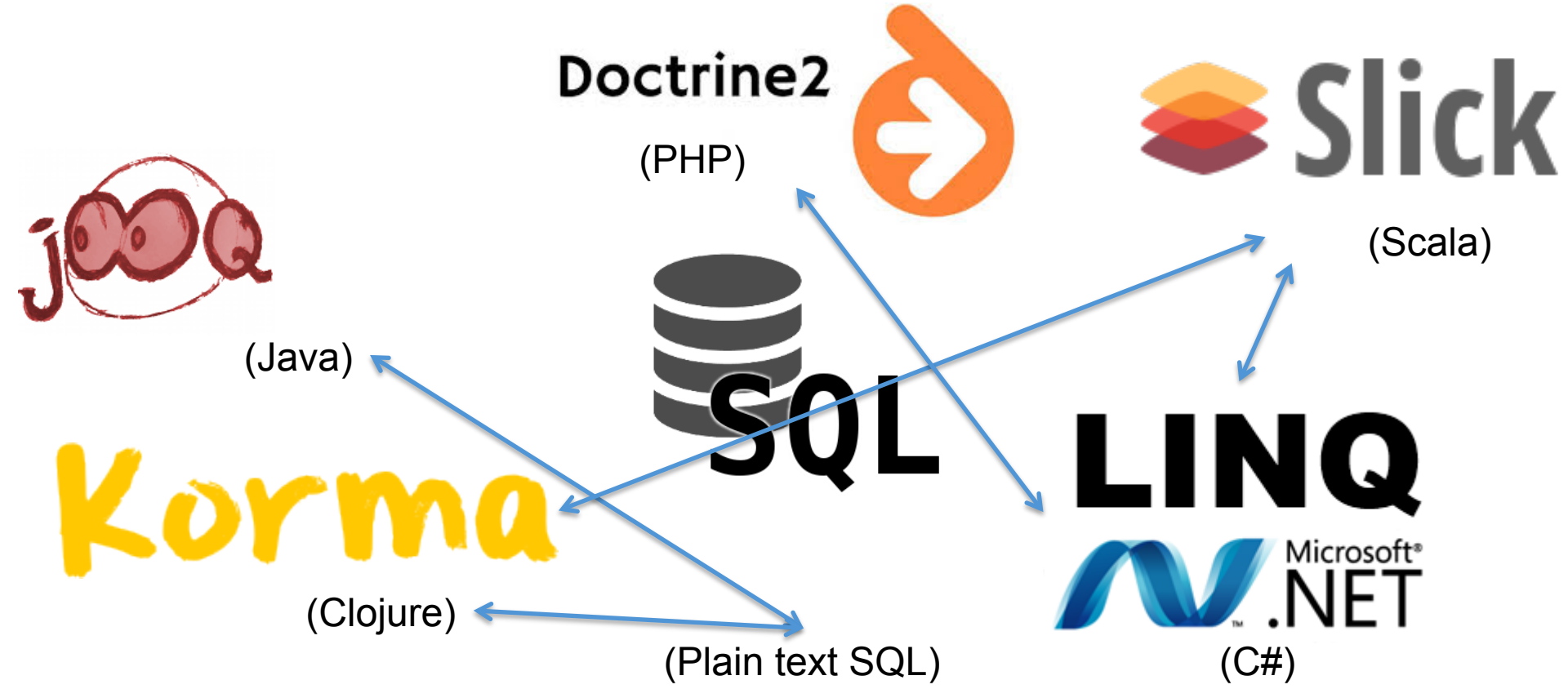
(Plain text SQL)

Notation and Services differ.
Pros. Cons. in both sides.

```java
Result<Record> result =
create.select()
      .from(T_AUTHOR.as("a"))
      .join(T_BOOK.as("b")).on(a.ID.equal(b.AUTHOR_ID))
      .where(a.YEAR_OF_BIRTH.greaterThan(1920)
      .and(a.FIRST_NAME.equal("Paulo")))
      .orderBy(b.TITLE)
      .fetch();
```

(Java)

Metamorphic DSLs:
moving from one shape to another



Plain SQL
(external DSL)

shape
#1

```sql
-- SQL
SELECT * FROM journal
   WHERE published_year = 2013
     AND publisher = 'IEEE'
ORDER BY title
```

Java
(internal DSL)

shape
#2

```
// JOOQ fluent API
ResultQuery q = create.selectFrom(JOURNAL)
                      .where(PUBLISHED_YEAR.equal(2013)
                      .and(PUBLISHER.equal("IEEE")))
                      .orderBy(TITLE);
```

Scala
(internal DSL)

shape
#3

```
journals
  .filter(journal => journal.published_year === 2013
       && journal.publisher === "IEEE")
  .sortBy(_.title)
```

Metamorphic DSL

Java
(SPLAR API)

shape
#0

```java
String featureModelPath = "c:\\users\\marcilio\\eclipse\\fmrlib\\resources\\fm_samples\\simple_bike_fm.xml";

// Create feature model object from an XML file (SXFM format - see www.splot-research.org for details)
// If an identifier is not provided for a feature use the feature name as id
FeatureModel featureModel = new XMLFeatureModel(featureModelPath, XMLFeatureModel.USE_VARIABLE_NAME_AS_ID);
// load feature model from
featureModel.loadModel();

// create BDD variable order heuristic
new FTPreOrderSortedECTraversalHeuristic("Pre-CL-MinSpan", featureModel, FTPreOrderSortedECTraversalHeuristic.FORCE_SORT);
VariableOrderingHeuristic heuristic = VariableOrderingHeuristicsManager.createHeuristicsManager().getHeuristic("Pre-CL-MinSpan");

// BDD construction parameters
// - Tuning this parameters can be tricky at times and may require playing a bit
// - For the purpose of this example let's assume "large enough" values
int bddNodeNum = 10000;         // sets the initial size of the BDD table
int bddCacheSize = 10000;  // sets the size of the BDD cache table

// Creates the BDD reasoner
ReasoningWithBDD reasoner = new FMReasoningWithBDD(featureModel, heuristic, 50000, 50000, 60000, "pre-order");

// Initialize the reasoner (BDD is created at this moment)
reasoner.init();

// Use the reasoner
System.out.println("BDD has " + reasoner.getBDD().nodeCount() + " nodes and was built in " + reasoner.getBDDBuildingTime() + " ms");

// Check if feature model is consistent, i.e., has at least one valid configuration
System.out.println("Feature model is " + (reasoner.isConsistent() ? "" : " NOT ") + "consistent!");

// Count feature model solutions
System.out.println("Feature model has " + reasoner.countValidConfigurations() + " possible configurations");
```

FAMILIAR
(external DSL)

shape
#1

```
fm1 = FM ("fm1.xml")
n1 = counting fm1
b1 = isValid fm1
fm2 = FM ("fm2.tvl")
fm3 = merge union { fm1 fm2 }
slice fm3 including { A E F }
```

Java
(internal DSL)

shape
#2

```java
@Test
public void example1() throws Exception {
    FeatureModelVariable fm1 = FM ("fm1.xml") ;
    double n1 = fm1.counting();
    boolean b1 = fm1.isValid();
    FeatureModelVariable fm2 = FM ("fm2.tvl");
    FeatureModelVariable fm3 = fm1.merge(fm2, Mode.Union);
    FeatureModelVariable fm4 = fm3.slice(SliceMode.INCLUDING, "A", "E", "F");
}
```

Scala
(internal DSL)

shape
#3

```scala
class FamiscaleExample extends FamiscaleScript {
  // your FAMISCALE code here
  var fm1 = FM("fm1.xml")
  var n1 = fm1.counting
  var b1 = fm1.isValid
  var fm2 = FM("fm2.tvl")
  var fm3 = merge union (fm1, fm2)
  var fm4 = slice(fm3 including ('A, 'E, 'F))
}
```

Metamorphic
DSL

Metamorphic DSL

Vision: software languages should be self-adaptable to the most appropriate shape (including the corresponding IDE) and according to a particular usage or task.

Plain SQL (external DSL)

shape #1

```sql
-- SQL
SELECT * FROM journal
   WHERE published_year = 2013
     AND publisher = 'IEEE'
ORDER BY title
```

Java (internal DSL)

shape #2

```java
// JOOQ fluent API
ResultQuery q = create.selectFrom(JOURNAL)
           .where(PUBLISHED_YEAR.equal(2013)
           .and(PUBLISHER.equal("IEEE")))
           .orderBy(TITLE);
```

Scala (internal DSL)

shape #3

```scala
journals
  .filter(journal => journal.published_year === 2013
      && journal.publisher === "IEEE")
  .sortBy(_.title)
```

Where the idea of metamorphic DSL comes from?

- **Analysis** of the DSL erea
  - Socio-technical aspects of DSLs suggest the idea of supporting different shapes
- 4-years **experience** of developping different shapes of a DSL
  - FAMILIAR (for performing operations over feature models)
- Scenarios with Metamorphic DSLs

Socio-Technical Aspects of DSLs

#1 Diversity of terminology

- Traditional dichotomy between internal DSL and external DSL (Fowler et al., 2010)
  - Fluent APIs
  - Internal DSLs
  - (deeply) embedded DSLs
  - External DSLs
  - What's LINQ?
- Boundary between DSL and GPL is not that clear (Voelter et al., 2013)
  - What is and what is not a DSL is still a debate

#1 The diversity of terminology shows the **large spectrum of shapes DSLs** can take

#2 Syntax and Environment Matter

- Promises of DSL« improvement » in terms of
  - usability, learnability, expressiveness, reusability, etc.
- Empirical study on the role of syntax
  - C-style syntax induces problems in terms of usability for novices; language more or less intuitive for (non-)programmers (Stefik et al. 2014)
  - Syntax issues with Java for students (Denny et al. 2011)
  - PL usability: method namings/placement, use of identifiers, API design (Ellis et al., Styllos et al., Clarke, Montperrus et al., etc.)
- More specialized/sophicated tools/IDE can be derived from a DSL
  - editors, compilers, debuggers

#2 As syntax and development environment matter, we should allow the **user to choose the right shape** of a DSL

Meyerovich and Rabkin « Empirical analysis of programming language adoption » OOPSLA'13

		Ensō	Más	MetaEdit+	MPS	Onion	Rascal	Spoofax	SugarJ	Whole	Xtext
Notation	Textual	●	●		●	●	●	●	●	●	●
	Graphical	●	◒	●			◒			●	
	Tabular		●	●	●					●	
	Symbols			●	●					●	
Semantics	Model2Text		●	●	●	●	●	●	●	●	●
	Model2Model			●	●	●	●	●	●	●	●
	Concrete syntax			●	●	●	●	●	●		
	Interpretative	●		●	●		◒	●		●	●
Validation	Structural	●	●	●	●	●	●	●	●	●	●
	Naming	◒	●	●	●	●		●		●	◒
	Types				●				●		●
	Programmatic	●			●	●	●	●	●		●
Testing	DSL testing				●		◒	●		●	●
	DSL debugging	●		●	●		●			●	●
	DSL prog. debugging	●			●					●	●
Composability	Syntax/views	●		●	●	●	●	●	●	●	◒
	Validation			●	●	●	●	●	●	●	●
	Semantics	●		●	●	●	●	●	●		●
	Editor services			●	●	●	●	●	●		●
Editing mode	Free-form	●		●		●	●	●	●		●
	Projectional		●		●	●				●	
Syntactic services	Highlighting		◒	●	●	●	●	●	●	●	●
	Outline			●	●	●	●	●	●	●	●
	Folding		●	●	●	●	●	●	●	●	●
	Syntactic completion			●	●	●		●	●		●
	Diff	●		●	●	●	●	●	●		●
	Auto formatting	●	●	●	●	●	●	●		●	●
Semantic services	Reference resolution		●	●	●	●	●	●	●		●
	Semantic completion		●	●	●	●	●	●	●	●	●
	Refactoring		◒	●	●		●	●		●	
	Error marking		●	●	●	●	●	●	●	●	●
	Quick fixes				●						●
	Origin tracking	●		●	●		●	●	●		●
	Live translation			●		●	◒	●		●	●

Table 1: Language Workbench Features (● = full support, ◒ = partial/limited support)

#3 Language Workbenches

Erdweg et al. SLE'13

#3 The community of language engineering is providing more and more mature solutions for building DSLs – being external or internal. **Developers** of DSLs have now a **variety of strategies to choose from and build an appropriate shape**.

Shaping Up DSL!

- Diversity of terminological clarifications
- Role of syntax and environment
- Developers can devise new DSL
- All suggest the idea of **having different shapes of a DSL**

- What is missing is a systematic solution for transitioning from one shape to another
  - We would like to open a given artefact (expressed in a DSL) with another syntax and another environment
  - **Metamorphic DSL**

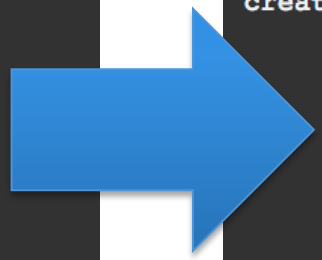
Putting All Together

Metamorphic DSL scenarios

Scenario #1

- Help people to quickly learn the language and transition to another shape when needs be
  - NB: can be the same person!

```
-- Select all books by authors born after 1920,
-- named "Paulo" from a catalogue:
SELECT *
  FROM t_author a
  JOIN t_book b ON a.id = b.author_id
 WHERE a.year_of_birth > 1920
   AND a.first_name = 'Paulo'
 ORDER BY b.title
```

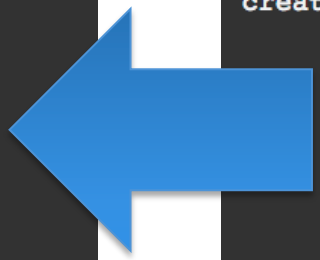


```
Result<Record> result =
create.select()
      .from(T_AUTHOR.as("a"))
      .join(T_BOOK.as("b")).on(a.ID.equal(b.AUTHOR_ID))
      .where(a.YEAR_OF_BIRTH.greaterThan(1920)
      .and(a.FIRST_NAME.equal("Paulo")))
      .orderBy(b.TITLE)
      .fetch();
```

Scenario #2

- People can use the more advanced support for understanding / debugging SQL queries
  - NB: can be the same person!

```
-- Select all books by authors born after 1920,
-- named "Paulo" from a catalogue:
SELECT *
  FROM t_author a
  JOIN t_book b ON a.id = b.author_id
 WHERE a.year_of_birth > 1920
   AND a.first_name = 'Paulo'
 ORDER BY b.title
```



```
Result<Record> result =
create.select()
      .from(T_AUTHOR.as("a"))
      .join(T_BOOK.as("b")).on(a.ID.equal(b.AUTHOR_ID))
      .where(a.YEAR_OF_BIRTH.greaterThan(1920)
      .and(a.FIRST_NAME.equal("Paulo")))
      .orderBy(b.TITLE)
      .fetch();
```

Scenario #(1+2)

- People can use the more advanced support for understanding / debugging SQL queries
  - And back again!

```
-- Select all books by authors born after 1920,
-- named "Paulo" from a catalogue:
SELECT *
  FROM t_author a
  JOIN t_book b ON a.id = b.author_id
 WHERE a.year_of_birth > 1920
   AND a.first_name = 'Paulo'
 ORDER BY b.title
```

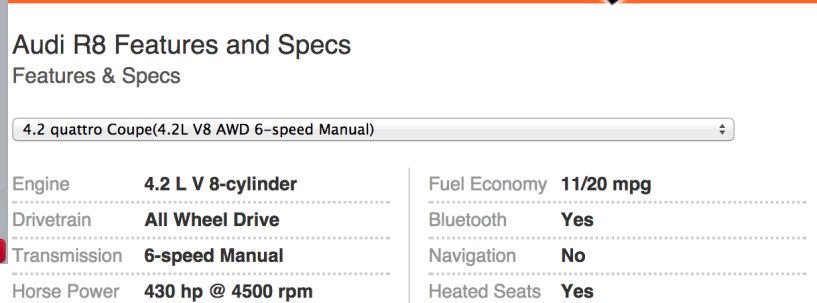
```
Result<Record> result =
create.select()
      .from(T_AUTHOR.as("a"))
      .join(T_BOOK.as("b")).on(a.ID.equal(b.AUTHOR_ID))
      .where(a.YEAR_OF_BIRTH.greaterThan(1920)
      .and(a.FIRST_NAME.equal("Paulo")))
      .orderBy(b.TITLE)
      .fetch();
```

Scenario #(Putting All Together)

Say a company wants to develop a **web configurator** for assisting customers in the selection of **products**

play

SQL



```
1  SELECT *
2  FROM products
3  WHERE gps = 'true'
```

```
mysql> SELECT * FROM products;
+-----------+-------------+-----------+----------+------------+
| productID | productCode | name      | quantity | gps        |
+-----------+-------------+-----------+----------+------------+
|      1001 | PEN         | Car Red   |     5000 |       true |
|      1002 | PEN         | Car Blue  |     8000 |       true |
|      1003 | PEN         | Car Black |     2000 |       true |
|      1004 | PEC         | Car 2B    |    10000 |       true |
|      1005 | PEC         | Car 2H    |     8000 |       false|
|      1006 | PEC         | Car HB    |        0 |       false|
+-----------+-------------+-----------+----------+------------+
6 rows in set (0.02 sec)
```

SQL

product manager

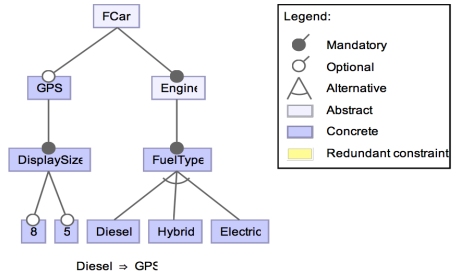
FAMILIAR

carFamily.fml

```
1  fmCarFamily = FM (FCar: [GPS] [Engine];
2                    GPS : DisplaySize ; DisplaySize : ("5"|"8") ;
3                    Engine : FuelType ; FuelType : (Diesel|Electric|Hybrid) ;
4                    Diesel -> GPS ; // business decision
5  )
6  c1 = configuration fmCarFamily
```

Console

```
FAMILIAR Console
FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Reasoning)  version 1.1
http://familiar-project.github.com/
fml> run "carFamily.fml"
fmCarFamily: (FEATURE_MODEL) FCar: [Engine] [GPS] ;
Engine: FuelType ;
GPS: DisplaySize ;
FuelType: (Diesel|Hybrid|Electric) ;
DisplaySize: ("8"|"5") ;
(Diesel -> GPS);
c1: (CONFIGURATION) selected: [FCar]       deselected: []
fml> select Diesel in c1
res0: (BOOLEAN) true
fml> selectedF c1
res1: (SET) {Engine;GPS;DisplaySize;FuelType;FCar;Diesel}
fml>
```



marketing engineer

We also have an Eclipse version

(Java API)

```java
@Test
public void example2() throws Exception {
    // handling the configuration logic
    FeatureModelVariable fmCarFamily = FM ("FCar: [GPS] [Engine]; "
            + "GPS : DisplaySize ; DisplaySize : (\"5\"|\"8\") ;"
            + "Engine : FuelType ; FuelType : (Diesel|Electric|Hybrid) ;"
            + " Diesel -> GPS ;"
    );
    ConfigurationVariable c1 = new ConfigurationVariableSPLOTImpl(fmCarFamily, "c1");
    //
    //
    //...
    if (c1.getSelected().contains("GPS")) {

        // querying the catalog of products
        Optional<Product> person =
                from(Product.class)
                    .where(Product::getGPS)
                    .equalTo("true")
                    .select(
                        productMapper,
                        connectionFactory::openConnection);
        //
        // display the information accordingly
        // ...

    }
}
```

software engineer

```java
@Test
public void example2() throws Exception {
    // handling the configuration logic
    FeatureModelVariable fmCarFamily = FM ("FCar: [GPS] [Engine]; "
            + "GPS : DisplaySize ; DisplaySize : (\"5\"|\"8\") ;"
            + "Engine : FuelType ; FuelType : (Diesel|Electric|Hybrid) ;"
            + " Diesel -> GPS ;"
    );
    ConfigurationVariable c1 = new ConfigurationVariableSPLOTImpl(fmCarFamily, "c1");
    //
    //
    //...
    if (c1.getSelected().contains("GPS")) {

        // querying the catalog of products
        Optional<Product> person =
                from(Product.class)
                    .where(Product::getGPS)
                    .equalTo("true")
                    .select(
                        productMapper,
                        connectionFactory::openConnection);
        //
        // display the information accordingly
        // ...

    }
}
```

FAMILIAR

(Java API)

jOOQ

SQL

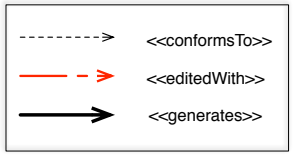
FAMILIAR

marketing engineer

software engineer

product manager

Metamorphic DSLs



SQL definition

FAMILIAR definition

a SQL query

a FAMILIAR script

```java
@Test
public void example2() throws Exception {
    // handling the configuration logic
    FeatureModelVariable fmCarFamily = FM ("FCar: [GPS] [Engine]; "
            + "GPS : DisplaySize ; DisplaySize : (\"5\"|\"8\") ;"
            + "Engine : FuelType ; FuelType : (Diesel|Electric|Hybrid) ;"
            + " Diesel -> GPS ;"
    ) ;
    ConfigurationVariable c1 = new ConfigurationVariableSPLOTImpl(fmCarFamily, "c1");
    //
    //
    //...
    if (c1.getSelected().contains("GPS")) {

        // querying the catalog of products
        Optional<Product> person =
                from(Product.class)
                    .where(Product::getGPS)
                    .equalTo("true")
                    .select(
                        productMapper,
                        connectionFactory::openConnection);
        //
        // display the information accordingly
        // ...

    }
}
```

```sql
SELECT *
FROM products
WHERE gps = 'true'
```

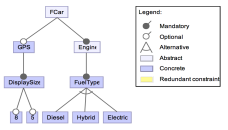
```
mysql> SELECT * FROM products;
+-----------+-------------+-----------+----------+------------+
| productID | productCode | name      | quantity | gps        |
+-----------+-------------+-----------+----------+------------+
|      1001 | PEN         | Car Red   |     5000 |       true |
|      1002 | PEN         | Car Blue  |     8000 |       true |
|      1003 | PEN         | Car Black |     2000 |       true |
|      1004 | PEC         | Car 2B    |    10000 |       true |
|      1005 | PEC         | Car 2H    |     8000 |      false |
|      1006 | PEC         | Car HB    |        0 |      false |
+-----------+-------------+-----------+----------+------------+
6 rows in set (0.02 sec)
```

software engineer

carFamily.fml

```
fmCarFamily = FM (FCar: [GPS] [Engine];
                  GPS : DisplaySize ; DisplaySize : ("5"|"8") ;
                  Engine : FuelType ; FuelType : (Diesel|Electric|Hybrid) ;
                  Diesel -> GPS ; // business decision
)
c1 = configuration fmCarFamily
```

```
FAMILIAR Console
FAMILIAR (for FeAture Model scRipt Language for manIpulation and Automatic Reasoning) version 1.1
http://familiar-project.github.com/
fml> run "carFamily.fml"
fmCarFamily: (FEATURE_MODEL) FCar: [Engine] [GPS] ;
Engine: FuelType ;
GPS: DisplaySize ;
FuelType: (Diesel|Hybrid|Electric) ;
DisplaySize: ("8"|"5") ;
(Diesel -> GPS);
c1: (CONFIGURATION) selected: [FCar]      deselected: []
fml> select Diesel in c1
res0: (BOOLEAN) true
fml> selectedF c1
res1: (SET) {Engine;GPS;DisplaySize;FuelType;FCar;Diesel}
fml>
```



product manager

marketing engineer

Research Directions

Research Directions

- What and where are metamorphic DSLs?
- Empirical studies for investigating the role of syntax and environment
- Solution for building metamorphic DSL
- Solution in the large: does metamorphic DSL pay off?

#1 Research Direction
(metamorphic classification)

- What are metamorphic DSLs?
  - Precise definition/scope is missing
- Where are metamorphic DSLs?
  - Empirical observation and inventory ongoing
- Are all DSLs metamorphic?

#2 Research Direction
(empirical studies)

- H0: Syntax and environment matter
  - H1: suboptimal notation can decrease understandibility
  - H2: non comprehensive environment support can drecrease usability or productivity
  - …
  - Hn: …
- But empirical evidence is missing
  - e.g., JOOQ for new learners <<<<< plain SQL for new learners?
  - e.g., Impact of the absence of profiling/benchmark tools

#3 Research Direction
(solution)

- Solution for building metamorphic DSL
  - Bi-directional transformations for moving from one shape to another (and back again)
  - Engineering a new shape of a DSL (when the shape is missing)
  - Reuse, « core »

- HCI issues
  - Copy and paste?
  - Open with?
  - Features of projectional editors?
  - Self-mechanism?

#4 Research Direction
(solution in the large)

- Engineering metamorphic DSLs
  - Does it pay off for users and developers?
    - From the users' perspective « switching » can have a cognitive cost
    - From the developers' perspective it requires effort (hopefully little!)
- Metamorphic scenarios
  - Communication/socio-technical issues
- Case studies, controlled experiments

Wrapping Up

Where the idea of Metamorphic DSL comes from?

- Languages with specialized notation and services exist because a one size-fits-all-solution is unlikely = DSLs
- Follow up argument: a one-size-fits-all solution is unlikely even for a given DSL (domain/class of problem)
  - Different shapes of a DSL
  - How to transition from one shape to another?
- Both users, developers, and researchers want to shape up DSLs
- All based on practical experience (FAMILIAR) and observation (e.g., literature, SQL)

Metamorphic DSL

- Optimistic view of DSL **diversity**
  - We should embrace, promote, and support diversity in DSLs ; users are diverse, have different requirements, and are using DSLs in numerous different contexts
- DSLs should be self-**adaptable to the most appropriate shape** (including the corresponding IDE), according to a particular usage or task

Metamorphic DSL

- Optimistic view of DSL **diversity**
- DSLs should be self-**adaptable to the most appropriate shape** (including the corresponding IDE), according to a particular usage or task
- A great future for DSL; additional research effort is needed
  - to further **understand** the concept of metamorphic DSL (what/where are they? why/when some shapes are more adequate?)
  - to provide effective **solution** for transitioning from one shape to another

Final ideas/remarks

DSLpedia.org ?

- Inventory of domain-specific languages
- Inventory of potential metamorphic domain-specific languages

The psychology of DSLs

COGNITIVE PSYCHOLOGY

How Language Shapes Thought

The languages we speak affect our perceptions of the world

By Lera Boroditsky

- “Even variations in grammar can profoundly affect how we see the world.”

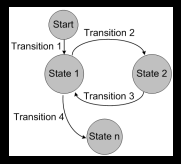
- DSLs: concrete syntax + tools
- Does syntax only matter?

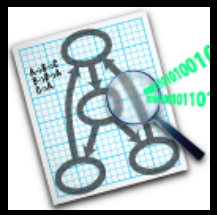
Metamorphic links

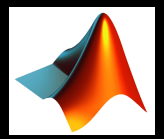
- Is is related to frictional forces?
- Is it related to product lines?

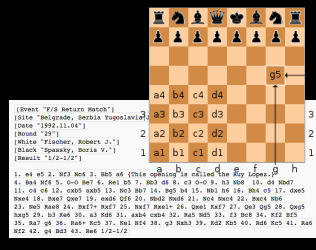
- Is it related to (M)SOS?

Do you want to play the metamorphic game?









The metamorphic game

Do all of the following DSLs have the potential to be (or even are) metamorphic?

HTML

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "DTD/xhtml1-transitional.dtd">
<html xml:lang="en" lang="en" xmlns="http://www.w3.org/1999/xhtml">
  <head>
      <title>Hello World</title>
  </head>
  <body>
      <p>My first Web page.</p>
  </body>
</html>
```

Domain: web (markup)

```
object XMLTest1 extends Application {
  val page =
  <html>
    <head>
      <title>Hello XHTML world</title>
    </head>
    <body>
      <h1>Hello world</h1>
      <p><a href="scala-lang.org">Scala</a> talks XHTML</p>
    </body>
  </html>;
  println(page.toString())
}
```

Scala

```
// Import the Glitter DSL
import glitter._

object Templates {

  // Define a reusable layout
  def layout(body: Xml) =
    html5dtd | 'html (
        'head :: 'title :: "Glitter is amazing!"
      | 'body :: body
    )

  // Define a template taking one String argument and using the layout defined above
  def show(name: String) =
    layout (
        'h1 :: "Show user"
      | 'p :: ("Hello " | 'strong(name) | "!")
    )

  // Define a template taking a List of Strings, using the layout defined above
  def index(users: List[String]) =
    layout (
        'h1 :: "User list"
      | 'ul % 'class~"user-list" :: (for (user <- users) yield ('li :: user))
    )
}
```

https://github.com/julienrf/glitter

TCS Wyvern (Omar et al., OOPLSA'14)

```
let webpage : HTML = HTMLElement(Dict.empty(), [BodyElement(Dict.empty(),
  [H1Element(Dict.empty(), [TextNode("Results for " + keyword)]),
  ULElement((Dict.add Dict.empty() ("id","results")), to_list_items(query(db,
    SelectStmt(["title", "snippet"], "products",
      [WhereClause(InPredicate(StringLit(keyword), "title"))]))))])])
```

```
let webpage : HTML = <html><body><h1>Results for {keyword}</h1>
  <ul id="results">{to_list_items(query(db,
    SELECT title, snippet FROM products WHERE {keyword} in title))}
  </ul></body></html>
```

```
let webpage : HTML = parse_html("<html><body><h1>Results for "+keyword+"</h1>
  <ul id=\"results\">" + to_string(to_list_items(query(db, parse_sql(
    "SELECT title, snippet FROM products WHERE '"+keyword+"' in title")))) +
  "</ul></body></html>")
```

Bibtex

Another metamorphic DSL?

https://github.com/inukshuk/bibtex-ruby

Using a Hash:

```
> bib << BibTeX::Entry.new({
    :bibtex_type => :book,
    :key => :rails,
    :address => 'Raleigh, North Carolina',
    :author => 'Ruby, Sam and Thomas, Dave, and Hansson, David Heinemeier',
    :booktitle => 'Agile Web Development with Rails',
    :edition => 'third',
    :keywords => 'ruby, rails',
    :publisher => 'The Pragmatic Bookshelf',
    :series => 'The Facets of Ruby',
    :title => 'Agile Web Development with Rails',
    :year => '2009'
  })
```

Or programmatically:

```
> book = BibTeX::Entry.new
> book.type = :book
> book.key = :mybook
> bib << book
```

CSS

```
.CodeMirror {
  line-height: 1;
  position: relative;
  overflow: hidden;
}

.CodeMirror-scroll {
  /* 30px is the magic margin used to hide the element's real scrollbars */
  /* See overflow: hidden in .CodeMirror, and the paddings in .CodeMirror-sizer */
  margin-bottom: -30px; margin-right: -30px;
  padding-bottom: 30px; padding-right: 30px;
  height: 100%;
  outline: none; /* Prevent dragging from highlighting the element */
  position: relative;
}
.CodeMirror-sizer {
  position: relative;
}
```

Domain: web (styling)

Makefile

```
PACKAGE      = package
    VERSION      = ` date "+%Y.%m%d%" `
    RELEASE_DIR  = ..
    RELEASE_FILE = $(PACKAGE)-$(VERSION)

    # Notice that the variable LOGNAME comes from the environment in
    # POSIX shells.
    #
    # target: all - Default target. Does nothing.
    all:
            echo "Hello $(LOGNAME), nothing to do by default"
            # sometimes: echo "Hello ${LOGNAME}, nothing to do by default"
            echo "Try 'make help'"

    # target: help - Display callable targets.
    help:
            egrep "^# target:" [Mm]akefile

    # target: list - List source files
    list:
            # Won't work. Each command is in separate shell
            cd src
            ls

            # Correct, continuation of the same shell
            cd src; \
            ls
```

Domain: software building

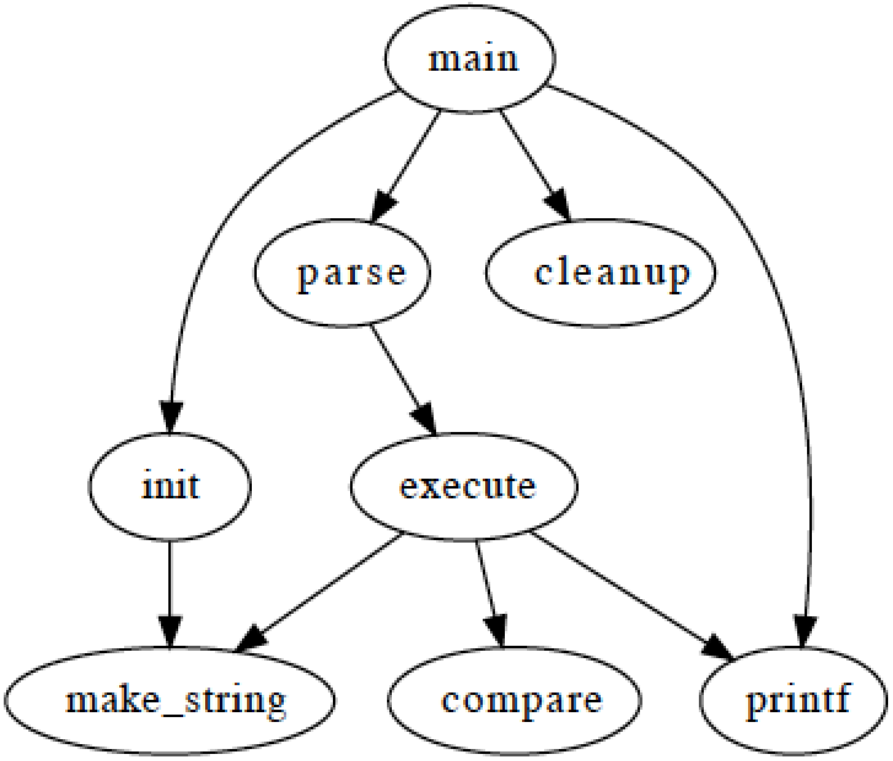
Lighthttpd configuration file

```
server.document-root = "/var/www/servers/www.example.org/pages/"

server.port = 80

server.username = "www"
server.groupname = "www"

mimetype.assign = (
  ".html" => "text/html",
  ".txt" => "text/plain",
  ".jpg" => "image/jpeg",
  ".png" => "image/png"
)

static-file.exclude-extensions = ( ".fcgi", ".php", ".rb", "~", ".inc" )
index-file.names = ( "index.html" )
```

Domain: web server (configuration)

Graphviz

```
digraph G {
main -> parse -> execute;
main -> init;
main -> cleanup;
execute -> make_string;
execute -> printf
init -> make_string;
main -> printf;
execute -> compare;
 }
```



Domain: graph (drawing)

```
1  import uk.co.turingatemyhamster.graphvizs.exec
2
3  val gOut = dot2dotG(Graph(false, GraphType.Digraph, Some("g"),
4    EdgeStatement("a1") -> "a2" -> "a3" -> "a4",
5    EdgeStatement("a1") -> "a4" ))
```

http://drdozer.github.io/graphviz-s/

http://turingatemyhamster.co.uk/blog/?p=157

https://github.com/glejeune/Ruby-Graphviz/

```
g = GraphViz::new( "G" )
g.node["shape"] = "ellipse"
g.node["color"] = "black"

g["color"] = "black"

c0 = g.add_graph( "cluster0" )
c0["label"] = "process #1"
c0["style"] = "filled"
c0["color"] = "lightgrey"
a0 = c0.add_nodes( "a0", "style" => "filled", "color" => "white" )
a1 = c0.add_nodes( "a1", "style" => "filled", "color" => "white" )
a2 = c0.add_nodes( "a2", "style" => "filled", "color" => "white" )
a3 = c0.add_nodes( "a3", "style" => "filled", "color" => "white" )
c0.add_edges( a0, a1 )
c0.add_edges( a1, a2 )
c0.add_edges( a2, a3 )

c1 = g.add_graph( "cluster1", "label" => "process #2" )
b0 = c1.add_nodes( "b0", "style" => "filled", "color" => "blue" )
b1 = c1.add_nodes( "b1", "style" => "filled", "color" => "blue" )
b2 = c1.add_nodes( "b2", "style" => "filled", "color" => "blue" )
b3 = c1.add_nodes( "b3", "style" => "filled", "color" => "blue" )
c1.add_edges( b0, b1 )
c1.add_edges( b1, b2 )
c1.add_edges( b2, b3 )

start = g.add_nodes( "start", "shape" => "Mdiamond" )
endn  = g.add_nodes( "end",   "shape" => "Msquare" )
```

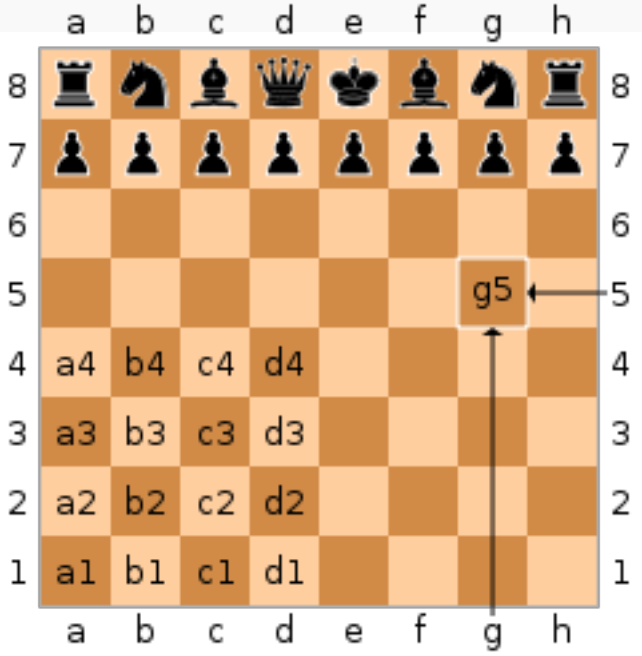
Regular expression

```
<TAG\b[^>]*>(.*?)</TAG>
```

Domain: strings (pattern matching)

Portable Game Notation (PGN)

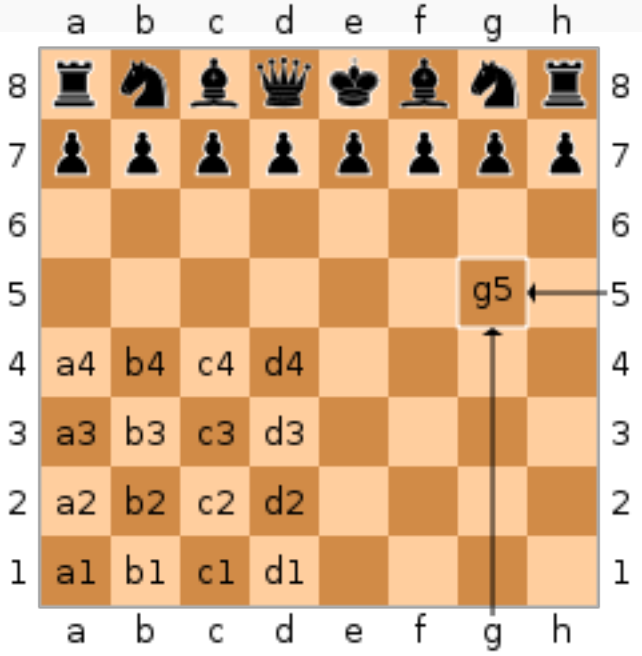
```
[Event "F/S Return Match"]
[Site "Belgrade, Serbia Yugoslavia|JUG"]
[Date "1992.11.04"]
[Round "29"]
[White "Fischer, Robert J."]
[Black "Spassky, Boris V."]
[Result "1/2-1/2"]

1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 {This opening is called the Ruy Lopez.}
4. Ba4 Nf6 5. O-O Be7 6. Re1 b5 7. Bb3 d6 8. c3 O-O 9. h3 Nb8  10. d4 Nbd7
11. c4 c6 12. cxb5 axb5 13. Nc3 Bb7 14. Bg5 b4 15. Nb1 h6 16. Bh4 c5 17. dxe5
Nxe4 18. Bxe7 Qxe7 19. exd6 Qf6 20. Nbd2 Nxd6 21. Nc4 Nxc4 22. Bxc4 Nb6
23. Ne5 Rae8 24. Bxf7+ Rxf7 25. Nxf7 Rxe1+ 26. Qxe1 Kxf7 27. Qe3 Qg5 28. Qxg5
hxg5 29. b3 Ke6 30. a3 Kd6 31. axb4 cxb4 32. Ra5 Nd5 33. f3 Bc8 34. Kf2 Bf5
35. Ra7 g6 36. Ra6+ Kc5 37. Ke1 Nf4 38. g3 Nxh3 39. Kd2 Kb5 40. Rd6 Kc5 41. Ra6
Nf2 42. g4 Bd3 43. Re6 1/2-1/2
```



PGN (Portable Game Notation)

Another metamorphic DSL coming from the chess domain

```
[Event "F/S Return Match"]
[Site "Belgrade, Serbia Yugoslavia|JUG"]
[Date "1992.11.04"]
[Round "29"]
[White "Fischer, Robert J."]
[Black "Spassky, Boris V."]
[Result "1/2-1/2"]

1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 {This opening is called the Ruy Lopez.}
4. Ba4 Nf6 5. O-O Be7 6. Re1 b5 7. Bb3 d6 8. c3 O-O 9. h3 Nb8  10. d4 Nbd7
11. c4 c6 12. cxb5 axb5 13. Nc3 Bb7 14. Bg5 b4 15. Nb1 h6 16. Bh4 c5 17. dxe5
Nxe4 18. Bxe7 Qxe7 19. exd6 Qf6 20. Nbd2 Nxd6 21. Nc4 Nxc4 22. Bxc4 Nb6
23. Ne5 Rae8 24. Bxf7+ Rxf7 25. Nxf7 Rxe1+ 26. Qxe1 Kxf7 27. Qe3 Qg5 28. Qxg5
hxg5 29. b3 Ke6 30. a3 Kd6 31. axb4 cxb4 32. Ra5 Nd5 33. f3 Bc8 34. Kf2 Bf5
35. Ra7 g6 36. Ra6+ Kc5 37. Ke1 Nf4 38. g3 Nxh3 39. Kd2 Kb5 40. Rd6 Kc5 41. Ra6
Nf2 42. g4 Bd3 43. Re6 1/2-1/2
```



PGN and DSL promises

```
[Event "F/S Return Match"]
[Site "Belgrade, Serbia Yugoslavia|JUG"]
[Date "1992.11.04"]
[Round "29"]
[White "Fischer, Robert J."]
[Black "Spassky, Boris V."]
[Result "1/2-1/2"]

1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 {This opening is called the Ruy Lopez.}
4. Ba4 Nf6 5. O-O Be7 6. Re1 b5 7. Bb3 d6 8. c3 O-O 9. h3 Nb8  10. d4 Nbd7
11. c4 c6 12. cxb5 axb5 13. Nc3 Bb7 14. Bg5 b4 15. Nb1 h6 16. Bh4 c5 17. dxe5
Nxe4 18. Bxe7 Qxe7 19. exd6 Qf6 20. Nbd2 Nxd6 21. Nc4 Nxc4 22. Bxc4 Nb6
23. Ne5 Rae8 24. Bxf7+ Rxf7 25. Nxf7 Rxe1+ 26. Qxe1 Kxf7 27. Qe3 Qg5 28. Qxg5
hxg5 29. b3 Ke6 30. a3 Kd6 31. axb4 cxb4 32. Ra5 Nd5 33. f3 Bc8 34. Kf2 Bf5
35. Ra7 g6 36. Ra6+ Kc5 37. Ke1 Nf4 38. g3 Nxh3 39. Kd2 Kb5 40. Rd6 Kc5 41. Ra6
Nf2 42. g4 Bd3 43. Re6 1/2-1/2
```

Easy to read, write, exchange, process

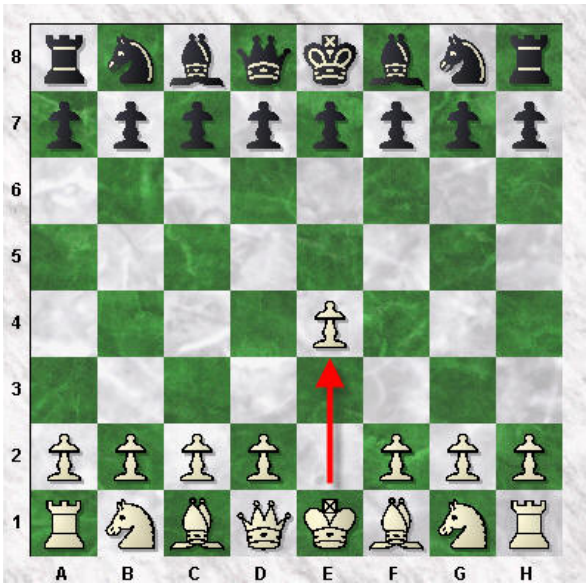
It is all about games information and moves (nothing more)

Success (chess players/tools): 8 millions games

« Alternatives » to PGN (other DSLs)

- For handling chess variants
  - e.g., Chess960
- Proprietary extensions
- For recording a particular game position
  - Forsyth–Edwards Notation (FEN)

rnbqkbnr/pppppppp/8/8/4P3/8/PPPP1PPP/
RNBQKBNR b KQkq e3 0 1



```ruby
require 'spec_helper'

describe PGN do
  describe "parsing a file" do
    it "should return a list of games" do
      games = PGN.parse(File.read("./examples/immortal_game.pgn"))
      games.length.should == 1
      game = games.first
      game.result.should == "1-0"
      game.tags["White"].should == "Adolf Anderssen"
      game.moves.last.should == "Be7#"
    end
  end
```

```
[Event "London"]
[Site "London"]
[Date "1851.??.??"]
[EventDate "?"]
[Round "?"]
[Result "1-0"]
[White "Adolf Anderssen"]
[Black "Kieseritzky"]
[ECO "C33"]
[WhiteElo "?"]
[BlackElo "?"]
[PlyCount "45"]

1.e4 e5 2.f4 exf4 3.Bc4 Qh4+ 4.Kf1 b5 5.Bxb5 Nf6 6.Nf3 Qh6 7.d3 Nh5 8.Nh4 Qg5
9.Nf5 c6 10.g4 Nf6 11.Rg1 cxb5 12.h4 Qg6 13.h5 Qg5 14.Qf3 Ng8 15.Bxf4 Qf6
16.Nc3 Bc5 17.Nd5 Qxb2 18.Bd6 Bxg1 19. e5 Qxa1+ 20. Ke2 Na6 21.Nxg7+ Kd8
22.Qf6+ Nxf6 23.Be7# 1-0
```

I have a question

Is it a domain-specific language!??!

```
require "spec_helper"

describe PGN::Game do
  describe "#positions" do
    it "should not raise an error" do
      tags   = {"White" => "Deep Blue", "Black" => "Kasparov"}
      moves  = %w{e4 c5 c3 d5 exd5 Qxd5 d4 Nf6 Nf3 Bg4 Be2 e6 h3 Bh5 O-O Nc6 Be3
      result = "1-0"
      game = PGN::Game.new(moves, tags, result)
      lambda { game.positions }.should_not raise_error
    end
  end
end
```

https://github.com/capicue/pgn

Is it a DSL?

YES, yes, yes !!!

Just another « shape » of PGN

```ruby
require "spec_helper"

describe PGN::Game do
  describe "#positions" do
    it "should not raise an error" do
      tags   = {"White" => "Deep Blue", "Black" => "Kasparov"}
      moves  = %w{e4 c5 c3 d5 exd5 Qxd5 d4 Nf6 Nf3 Bg4 Be2 e6 h3 Bh5 O-O Nc6 Be3
      result = "1-0"
      game = PGN::Game.new(moves, tags, result)
      lambda { game.positions }.should_not raise_error
    end
  end
end
```

FEN and PGN in Ruby

```
3  describe PGN::Position do
4    context "disambiguating moves" do
5      describe "using SAN square disambiguation" do
6        pos = PGN::FEN.new("r1bqkb1r/pp1p1ppp/2n1pn2/8/3NP3/2N5/PPP2PPP/R1BQKB1R w KQkq - 3 6").to_position
7        next_pos = pos.move("Ndb5")
8
9        it "should move the specified piece" do
10         next_pos.board.at("d4").should be_nil
11       end
12
13       it "should not move the other piece" do
14         next_pos.board.at("c3").should == "N"
15       end
16     end
```

```python
import unittest
import pgn

def game_fixture():
    game = pgn.PGNGame(
        'F/S Return Match',
        'Belgrade, Serbia Yugoslavia|JUG',
        '1992.11.04',
        '29',
        'Fischer, Robert J.',
        'Spassky, Boris V.',
        '1/2-1/2'
    )

    game.annotator = 'Renato'
    game.plycount = '3'
    game.timecontrol = '40/7200:3600'
    game.time = '12:32:43'
    game.termination = 'abandoned'
    game.mode = 'ICS'
    game.fen = 'rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/R1BQKBNR'

    game.moves = ['e4', 'e5', 'd4', 'd5', 'f3', '1/2-1/2']

    return game

PGN_TEXT = '''[Event "F/S Return Match"]
[Site "Belgrade, Serbia Yugoslavia|JUG"]
[Date "1992.11.04"]
[Round "29"]
[White "Fischer, Robert J."]
[Black "Spassky, Boris V."]
[Result "1/2-1/2"]
[Annotator "Renato"]
[PlyCount "3"]
[TimeControl "40/7200:3600"]
[Time "12:32:43"]
[Termination "abandoned"]
[Mode "ICS"]
[FEN "rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/R1BQKBNR"]

1. e4 e5 2. d4 d5 3. f3 1/2-1/2'''
```

PGN in Python?

https://github.com/renatopp/pgnparser

```python
import unittest
import pgn

def game_fixture():
    game = pgn.PGNGame(
        'F/S Return Match',
        'Belgrade, Serbia Yugoslavia|JUG',
        '1992.11.04',
        '29',
        'Fischer, Robert J.',
        'Spassky, Boris V.',
        '1/2-1/2'
    )

    game.annotator = 'Renato'
    game.plycount = '3'
    game.timecontrol = '40/7200:3600'
    game.time = '12:32:43'
    game.termination = 'abandoned'
    game.mode = 'ICS'
    game.fen = 'rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/R1BQKBNR'

    game.moves = ['e4', 'e5', 'd4', 'd5', 'f3', '1/2-1/2']

    return game
```

```
[Event "London"]
[Site "London"]
[Date "1851.??.??"]
[EventDate "?"]
[Round "?"]
[Result "1-0"]
[White "Adolf Anderssen"]
[Black "Kieseritzky"]
[ECO "C33"]
[WhiteElo "?"]
[BlackElo "?"]
[PlyCount "45"]

1.e4 e5 2.f4 exf4 3.Bc4 Qh4+ 4.Kf1 b5 5.Bxb5 Nf6 6.Nf3 Qh6 7.d3 Nh5 8.Nh4 Qg5
9.Nf5 c6 10.g4 Nf6 11.Rg1 cxb5 12.h4 Qg6 13.h5 Qg5 14.Qf3 Ng8 15.Bxf4 Qf6
16.Nc3 Bc5 17.Nd5 Qxb2 18.Bd6 Bxg1 19. e5 Qxa1+ 20. Ke2 Na6 21.Nxg7+ Kd8
22.Qf6+ Nxf6 23.Be7# 1-0
```

```ruby
require "spec_helper"

describe PGN::Game do
  describe "#positions" do
    it "should not raise an error" do
      tags   = {"White" => "Deep Blue", "Black" => "Kasparov"}
      moves  = %w{e4 c5 c3 d5 exd5 Qxd5 d4 Nf6 Nf3 Bg4 Be2 e6 h3 Bh5 O-O Nc6 Be3
      result = "1-0"
      game = PGN::Game.new(moves, tags, result)
      lambda { game.positions }.should_not raise_error
    end
  end
end
```

PGN in

????

```python
import unittest
import pgn

def game_fixture():
    game = pgn.PGNGame(
        'F/S Return Match',
        'Belgrade, Serbia Yugoslavia|JUG',
        '1992.11.04',
        '29',
        'Fischer, Robert J.',
        'Spassky, Boris V.',
        '1/2-1/2'
    )

    game.annotator = 'Renato'
    game.plycount = '3'
    game.timecontrol = '40/7200:3600'
    game.time = '12:32:43'
    game.termination = 'abandoned'
    game.mode = 'ICS'
    game.fen = 'rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/R1BQKBNR'

    game.moves = ['e4', 'e5', 'd4', 'd5', 'f3', '1/2-1/2']

    return game
```

```ruby
require "spec_helper"

describe PGN::Game do
  describe "#positions" do
    it "should not raise an error" do
      tags   = {"White" => "Deep Blue", "Black" => "Kasparov"}
      moves  = %w{e4 c5 c3 d5 exd5 Qxd5 d4 Nf6 Nf3 Bg4 Be2 e6 h3 Bh5 O-O Nc6 Be3
      result = "1-0"
      game = PGN::Game.new(moves, tags, result)
      lambda { game.positions }.should_not raise_error
    end
  end
end
```

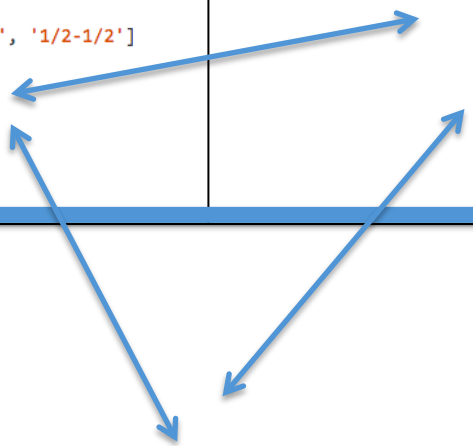
```
[Event "London"]
[Site "London"]
[Date "1851.??.??"]
[EventDate "?"]
[Round "?"]
[Result "1-0"]
[White "Adolf Anderssen"]
[Black "Kieseritzky"]
[ECO "C33"]
[WhiteElo "?"]
[BlackElo "?"]
[PlyCount "45"]

1.e4 e5 2.f4 exf4 3.Bc4 Qh4+ 4.Kf1 b5 5.Bxb5 Nf6 6.Nf3 Qh6 7.d3 Nh5 8.Nh4 Qg5
9.Nf5 c6 10.g4 Nf6 11.Rg1 cxb5 12.h4 Qg6 13.h5 Qg5 14.Qf3 Ng8 15.Bxf4 Qf6
16.Nc3 Bc5 17.Nd5 Qxb2 18.Bd6 Bxg1 19. e5 Qxa1+ 20. Ke2 Na6 21.Nxg7+ Kd8
22.Qf6+ Nxf6 23.Be7# 1-0
```

Internal Shapes
External Shapes

```python
import unittest
import pgn

def game_fixture():
    game = pgn.PGNGame(
        'F/S Return Match',
        'Belgrade, Serbia Yugoslavia|JUG',
        '1992.11.04',
        '29',
        'Fischer, Robert J.',
        'Spassky, Boris V.',
        '1/2-1/2'
    )

    game.annotator = 'Renato'
    game.plycount = '3'
    game.timecontrol = '40/7200:3600'
    game.time = '12:32:43'
    game.termination = 'abandoned'
    game.mode = 'ICS'
    game.fen = 'rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/R1BQKBNR'

    game.moves = ['e4', 'e5', 'd4', 'd5', 'f3', '1/2-1/2']

    return game

PGN_TEXT = '''[Event "F/S Return Match"]
[Site "Belgrade, Serbia Yugoslavia|JUG"]
[Date "1992.11.04"]
```

```ruby
require "spec_helper"

describe PGN::Game do
  describe "#positions" do
    it "should not raise an error" do
      tags   = {"White" => "Deep Blue", "Black" => "Kasparov"}
      moves  = %w{e4 c5 c3 d5 exd5 Qxd5 d4 Nf6 Nf3 Bg4 Be2 e6 h3 Bh5 O-O Nc6 Be3
      result = "1-0"
      game = PGN::Game.new(moves, tags, result)
      lambda { game.positions }.should_not raise_error
    end
  end
end
```

Internal Shapes (DSLs)



```
[Event "London"]
[Site "London"]
[Date "1851.??.??"]
[EventDate "?"]
[Round "?"]
[Result "1-0"]
[White "Adolf Anderssen"]
[Black "Kieseritzky"]
[ECO "C33"]
[WhiteElo "?"]
[BlackElo "?"]
[PlyCount "45"]

1.e4 e5 2.f4 exf4 3.Bc4 Qh4+ 4.Kf1 b5 5.Bxb5 Nf6 6.Nf3 Qh6 7.d3 Nh5 8.Nh4 Qg5
9.Nf5 c6 10.g4 Nf6 11.Rg1 cxb5 12.h4 Qg6 13.h5 Qg5 14.Qf3 Ng8 15.Bxf4 Qf6
16.Nc3 Bc5 17.Nd5 Qxb2 18.Bd6 Bxg1 19. e5 Qxa1+ 20. Ke2 Na6 21.Nxg7+ Kd8
22.Qf6+ Nxf6 23.Be7# 1-0
```

External Shapes (DSLs)

```python
import unittest
import pgn

def game_fixture():
    game = pgn.PGNGame(
        'F/S Return Match',
        'Belgrade, Serbia Yugoslavia|JUG',
        '1992.11.04',
        '29',
        'Fischer, Robert J.',
        'Spassky, Boris V.',
        '1/2-1/2'
    )

    game.annotator = 'Renato'
    game.plycount = '3'
    game.timecontrol = '40/7200:3600'
    game.time = '12:32:43'
    game.termination = 'abandoned'
    game.mode = 'ICS'
    game.fen = 'rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/R1BQKBNR'

    game.moves = ['e4', 'e5', 'd4', 'd5', 'f3', '1/2-1/2']

    return game

PGN_TEXT = '''[Event "F/S Return Match"]
[Site "Belgrade, Serbia Yugoslavia|JUG"]
[Date "1992.11.04"]
```

```ruby
require "spec_helper"

describe PGN::Game do
  describe "#positions" do
    it "should not raise an error" do
      tags   = {"White" => "Deep Blue", "Black" => "Kasparov"}
      moves  = %w{e4 c5 c3 d5 exd5 Qxd5 d4 Nf6 Nf3 Bg4 Be2 e6 h3 Bh5 O-O Nc6 Be3
      result = "1-0"
      game = PGN::Game.new(moves, tags, result)
      lambda { game.positions }.should_not raise_error
    end
  end
end
```

Metamorphic DSL

```
[Event "London"]
[Site "London"]
[Date "1851.??.??"]
[EventDate "?"]
[Round "?"]
[Result "1-0"]
[White "Adolf Anderssen"]
[Black "Kieseritzky"]
[ECO "C33"]
[WhiteElo "?"]
[BlackElo "?"]
[PlyCount "45"]

1.e4 e5 2.f4 exf4 3.Bc4 Qh4+ 4.Kf1 b5 5.Bxb5 Nf6 6.Nf3 Qh6 7.d3 Nh5 8.Nh4 Qg5
9.Nf5 c6 10.g4 Nf6 11.Rg1 cxb5 12.h4 Qg6 13.h5 Qg5 14.Qf3 Ng8 15.Bxf4 Qf6
16.Nc3 Bc5 17.Nd5 Qxb2 18.Bd6 Bxg1 19. e5 Qxa1+ 20. Ke2 Na6 21.Nxg7+ Kd8
22.Qf6+ Nxf6 23.Be7# 1-0
```

Backup slides

(part of MDE course in Rennes)

External DSLs vs Internal DSLs

- An **external** DSL is a completely separate language and has its own custom syntax/ tooling support (e.g., editor)

- An internal DSL is more or less a set of APIs written on top of a host language (e.g., Java).
  - Fluent interfaces

External vs Internal DSL (SQL example)

```
-- Select all books by authors born after 1920,
-- named "Paulo" from a catalogue:
SELECT *
  FROM t_author a
  JOIN t_book b ON a.id = b.author_id
 WHERE a.year_of_birth > 1920
   AND a.first_name = 'Paulo'
 ORDER BY b.title
```

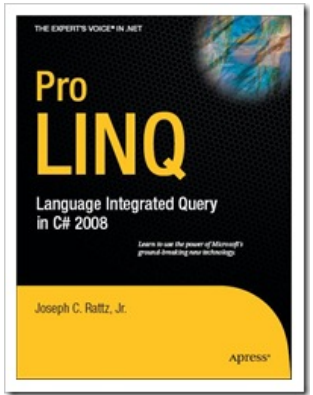
```
Result<Record> result =
create.select()
      .from(T_AUTHOR.as("a"))
      .join(T_BOOK.as("b")).on(a.ID.equal(b.AUTHOR_ID))
      .where(a.YEAR_OF_BIRTH.greaterThan(1920)
      .and(a.FIRST_NAME.equal("Paulo")))
      .orderBy(b.TITLE)
      .fetch();
```

DSL (LINQ/C# example)

```
// DataContext takes a connection string
DataContext db = new   DataContext("c:\\northwind\\northwnd.mdf");
// Get a typed table to run queries
Table<Customer> Customers = db.GetTable<Customer>();
// Query for customers from London
var q =
   from c in Customers
   where c.City == "London"
   select c;
foreach (var cust in q)
   Console.WriteLine("id = {0}, City = {1}", cust.CustomerID, cust.City);
```

LINQ

Microsoft® .NET



Internal DSL

- « Using a host language (e.g., Java) to give the host language the feel of a particular language. »
- **Fluent** Interfaces
  - « The more the use of the API has that language like flow, the more fluent it is »

```
-- Select all books by authors born after 1920,
-- named "Paulo" from a catalogue:
SELECT *
  FROM t_author a
  JOIN t_book b ON a.id = b.author_id
 WHERE a.year_of_birth > 1920
   AND a.first_name = 'Paulo'
 ORDER BY b.title
```

```
Result<Record> result =
create.select()
      .from(T_AUTHOR.as("a"))
      .join(T_BOOK.as("b")).on(a.ID.equal(b.AUTHOR_ID))
      .where(a.YEAR_OF_BIRTH.greaterThan(1920)
      .and(a.FIRST_NAME.equal("Paulo")))
      .orderBy(b.TITLE)
      .fetch();
```

SQL in… Java

DSL in GPL

```
Connection con = null;

// create sql insert query
String query = "insert into user values(" + student.getId() + ",'"
  + student.getFirstName() + "','" + student.getLastName()
  + "','" + student.getEmail() + "','" + student.getPhone()
  + "')";
try {
 // get connection to db
 con = new CreateConnection().getConnection("checkjdbc", "root",
   "root");

 // get a statement to execute query
 stmt = con.createStatement();

 // executed insert query
 stmt.execute(query);
 System.out.println("Data inserted in table !");
```

Regular expression in… Java

DSL in GPL

```
public class RegexTestStrings {
  public static final String EXAMPLE_TEST = "This is my small example "
      + "string which I'm going to " + "use for pattern matching.";

  public static void main(String[] args) {
    System.out.println(EXAMPLE_TEST.matches("\\w.*"));
    String[] splitString = (EXAMPLE_TEST.split("\\s+"));
    System.out.println(splitString.length);// Should be 14
    for (String string : splitString) {
      System.out.println(string);
    }
    // Replace all whitespace with tabs
    System.out.println(EXAMPLE_TEST.replaceAll("\\s+", "\t"));
  }
}
```

Internal DSLs vs External DSL

- Both internal and external DSLs have strengths and weaknesses (Fowler)
  - learning curve,
  - cost of building,
  - programmer familiarity,
  - communication with domain experts,
  - mixing in the host language,
  - strong expressiveness boundary
- Focus of the course
  - **external DSL** a completely separate language with its own custom syntax and tooling support (e.g., editor)

P e2 – e4

p g7 – g5

Knight at b2 moves to c3

pawn at f7 moves to f5

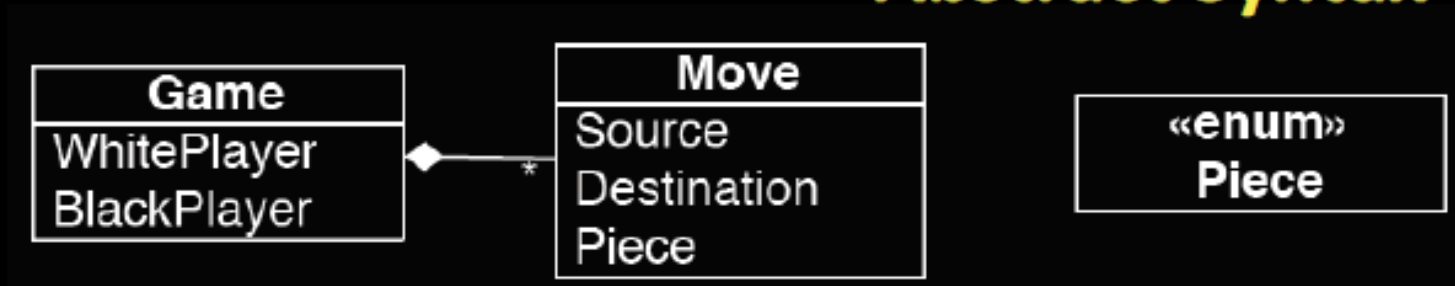
Q d1 – h5

1-0

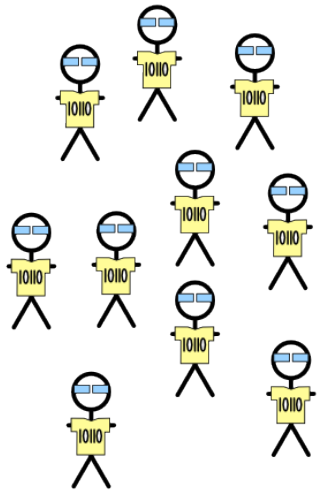
Concrete Syntax

Constraints !!!

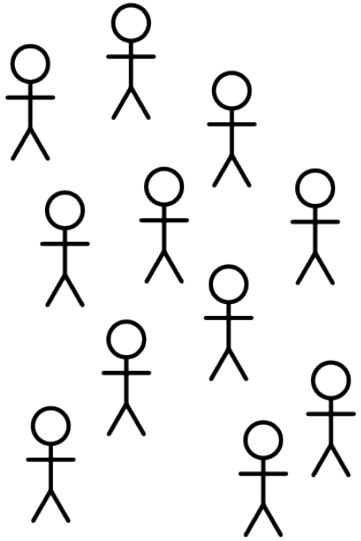
Abstract Syntax



Actors



Developers



End-Users