LexiFi: Approche “langage” pour la description de contrats financiers

Alain Frisch, LexiFi, directeur technique

Méthodes formelles et langages pour le développement de logiciels fiables dans l'industrie
28 avril 2014, IHP, Paris

Cet exposé

1. Approche "langage" pour la description de contrats financiers.
2. LexiFi et OCaml.

Contrats financiers

- Grande variété de produits financiers, parfois assez complexes (ça revient à la mode).
- Term-sheet: document légal définissant les termes et conditions d'un contrat financier; il régit les paiments (et livraisons d'autres instruments) entre deux parties contractantes, en fonction (i) d'observations sur les marchés et (ii) de décisions prises par les parties.

Exemple: Autocall worst-of sur 3 actions

- On observe le niveau initial de trois actions le 1er janvier 2014.
- Ensuite, à chacune des dates anniversaires pendant 5 ans, on observe la performance de chaque action par rapport à son niveau initial, et on considère X = la plus mauvaise des trois. 3 cas:
  - *0% <= X < 70%: on passe directement à la date suivante.*
  - *70% <= X < 100%: le produit paie un coupon de 50 euros et on passe à la date suivante.*
  - *100% <= X: le produit s'arrête en payant ce coupon de 50 euros ainsi que le "principal" de 1000 euros.*
- À la date finale, 2 cas:
  - *Si aucune des actions n'est tombée en dessous de 70% de son niveau initial pendant toute la vie du produit, le produit paie 1000 euros.*
  - *Sinon il paie 1000 * (1 + X) euros où X est la plus mauvaise des performances finales.*

Défis

- Les term sheets contiennent toujours au moins une erreur.
- Les institutions financières doivent executer des traitements variés sur ces contrats: valorisation, analyse de risques, gestion opérationnelle, reporting. Et cela pendant toute la vie du produit.
- De petites équipes gèrent des centaines/milliers de contrats un peu exotiques, bien souvent avec uniquement Excel + PDFs + post-its.
- Les systèmes informatiques qui gèrent ces contrats ignorent leur sémantique (syndrome "base relationnelle"), ou alors ils nécessitent des développements lourds pour chaque nouvelle structure et chaque nouveau traitement.
- De plus: cloisonnement entre classes d'actifs (sous-jacents) et métiers.
- La solution est évidente: il faut introduire une représentation uniforme et mécanisable, i.e. un langage informatique de description des contrats financiers.

Le langage de description de contrats de LexiFi

- Premières idées publiées en 2000: *Composing Contracts: An Adventure in Financial Engineering* (Simon Peyton Jones, Jean-Marc Eber, and Julian Seward). Jean-Marc a fondé LexiFi dans la foulée.
- Approche algébrique: quelques briques de bases et des combinateurs, avec une sémantique bien définie.
- Assez expressif pour décrire la plupart des produits financiers.
- Assez limité pour permettre d'automatiser de nombreux traitements et analyses.
- Approche stratifiée: les termes du DSL représentent des contrats complètement instanciés; ils sont habituellement générés à partir de paramètres par des "instruments" qui peuvent être exprimés dans n'importe langage.

Premier exemple

- Deux types de sous-termes:
  - `contract`: *un terme décrivant des paiements et livraisons;*
  - `'a value` *('a = date, float, bool, ...): la description d'un calcul pur.*
- Exemple:

```
cash_flow (cst 2020-06-30) (currency_of_string "USD") (cst 1000.)
```

- Combinateurs utilisés:

```
val cash_flow: date value -> currency -> float value -> contract
val currency_of_string: string -> currency
val cst: 'a -> 'a value
```

Quelques autres combinateurs

```
val all: contract list -> contract
val observe: string -> date value -> float value
val ifv: bool value -> 'a thing -> 'a thing -> 'a thing
val barrier: string -> date value -> date value -> (date value -> bool value) ->
      hit:(date value -> 'a thing) -> not_hit:'a thing -> 'a thing
val ( +.~ ): float value -> float value -> float value  (* infix op *)
```

- Exemple: option d'achat à prix fixé, soumise à condition activante:

```
  let cur = currency_of_string "USD" in
  let underlying = observe "MyStock Equity" in
  let strike = cst 4000. in
  let amount = max~ (cst 0.) (underlying (cst 2020-01-05) -.~ strike) in
  ifv (cst 3000. <=~ underlying (cst 2015-06-30))
    (cash_flow (cst 2020-01-05) cur amount)
    (all [])
```

Techniques "langages"

- Approche "locally nameless" pour la représentation des variables (Debruijn pour variables liées, identificateurs frais pour variables libres).
- Repésentation sous forme de DAG, avec partage maximal (hash-consing).
- Normalisation/évaluation lors de l'application des constructeurs (seuls des termes normalisés sont manipulés).
- Les événements externes (observations / décisions) donnent lieu à des réecritures des termes: le contrat "résiduel" est toujours représenté dans le même langage.
- Analyse statique (simple) pour obtenir une approximation de l'échéancier du contrat: dates des observations, date et montant des paiements, etc.
- Evaluation partielle des contrats pour exécution de scénarios, rapports graphiques, etc.
- Compilation vers code natif et/ou bytecode des contrats vers des formes de bas niveau pour faire du pricing efficace.

Le langage de description au sein des produits LexiFi

- L'implémentation du langage de description constitue le noyau de notre produit *LexiFi Apropos* destiné aux utilisateurs finaux (avec GUI et base de données), et est également embarquée en marque blanche dans des applications tierces (*Instrument Box)*.
- Apport principal: cette approche permet d'ajouter le support complet d'une nouvelle famille de produits en quelques jours; les nouvelles opérations (type analyse de risques) s'appliquent sur tous les produits existants (si elles peuvent être exprimées de manière générique, ce qui est souvent le cas).
- Demo...

LexiFi et OCaml

- LexiFi est un des utilisateurs industriels "historiques" d'OCaml (depuis 2001), membre du Consortium depuis qu'il existe.
- L'essentiel de notre base de code est en OCaml. (Lien avec .Net sous Windows: communication avec Excel, widgets, etc. Quelques routines numériques en C / Fortran.)
- En dehors du langage de description de contrats, plusieurs autres "sous-langages":
  - *langages intermédiaires pour la compilation "pricing",*
  - *algèbre de scénarios,*
  - *langage de templates pour la génération de documents,*
  - *"formules" dans les paramètres d'instruments.*

LexiFi et OCaml

- Nous contribuons activement à l'évolution du langage. Lorsque cela est nécessaire, nous maintenons quelques modifications locales.
- La principale modification non intégrée dans OCaml est un sous-système de représentation dynamique de la structure des types (=> programmation générique, génération automatique de GUI et de schémas SQL/XML, etc).
- Quelques exemples de contributions directes: dynlink natif, modules de première classe, warnings "qualité de code", "-ppx/extension points".
- Sous forme de financement: GADT, "-bin-annot".
- Nous redistribuons notre version du compilateur au sein de nos outils pour permettre de les étendre avec de nouveaux instruments. Certains de nos clients deviennent des utilisateurs d'OCaml! (Et deux d'entre eux ont adhéré au Consortium.)

Questions?