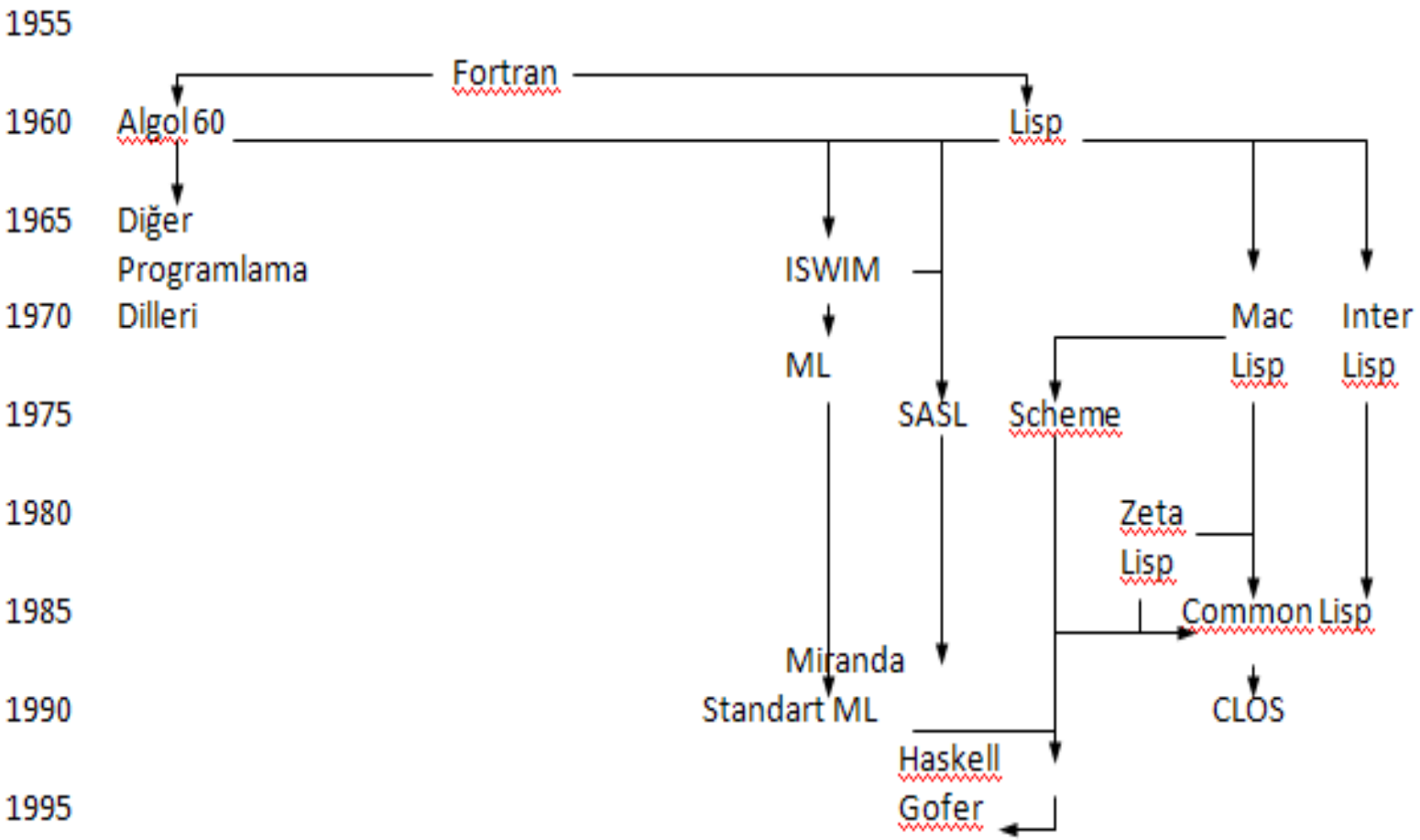
Fonksiyonel Programlama

LISP, ML, Haskel, Scheme

- Emir esaslı dillerin tasarımı doğrudan doğruya von Neumann mimarisine dayanır.
- Bir imperative dilde, işlemler yapılır ve sonuçlar daha sonra kullanım için değişkenlerde(variables) tutulur. Emir esaslı dillerde değişkenlerin yönetimi karmaşıklığa yol açar.
- Fonksiyonel dillerin tasarımı Matematiksel Fonksiyonlara dayalıdır ve değişkenler(variables), matematikte olduğu gibi gerekli değildir. Kullanıcıya da yakın olan sağlam bir teorik temele sahiptir.
- Fonksiyonel programlamada , bir fonksiyon aynı parametreler verildiğinde daima aynı sonucu üretir (referential transparency).

- Fonksiyonel dillerde problemin nasıl çözüleceğinden çok problemin ne olduğu önemlidir.
- For, if, while gibi denetim mekanizmaları makrolar halinde sunulur ve özyineleme ile gerçekleştirilir. Daha çok yapay zeka ve benzetim uygulamaları için uygun olabilir.
- Fonksiyon yaklaşımından dolayı matematik temeli oldukça sağlam olacağından optimize edilme (en iyileme) şansı çok yüksektir.

- Fonksiyonel programlama paradigması, Programlama dilini fonksiyon tanımının temel biçimleri üzerine oturtarak, algoritmaların ifadesi için basit ve açık bir ortam elde etmeyi amaçlamıştır.
- İlk örnek LISP dilidir ve onu ML, Scheme ve Haskell, dilleri izlemiştir.


1955
1960
1965
1970
1975
1980
1985
1990
1995
Fortran
Algol 60
Lisp
Diğer
Programlama
Dilleri
ISWIM
ML
SASL
Scheme
Mac
Lisp
Inter
Lisp
Zeta
Lisp
Common Lisp
Miranda
Standart ML
CLOS
Haskell
Gofer

- Sadece fonksiyonlar üzerine kurulmuş bir modeldir.
- Fonksiyonlar bir çok değer alır ve geriye sadece bir değer döndürürler.
- Fonksiyonlar başka fonksiyonları çağırır ya da başka fonksiyonun parametresi olur. Fonskiyonn(..(fonksiyon2(fonksiyon1(veriler)))..)
- Bu dillerde, alt yordamlar,fonksiyonlar (prosedürler) kullanılarak program daha alt parçalara bölünür.

Örnek:

- factorial (n) = if (n=0) then 1 else (n * factorial(n-1))
- (define factorial
- (lambda (n)
- (if (zero ? n)
- 1
- (* n (- n 1)))))

- Lisp dili Sembolik veri işleme amacı ile dizayn edilmiştir. Bu dil türev ve integral hesaplamalarındaki, elektrik devre teorisideki , matematiksel mantık oyunlarındaki ve yapay zekanın diğer alanlarındaki sembolik hesaplamalarda kullanılmaktadır. Karmaşık hesaplamalar daha basit ifadeler cinsinden yazılarak kolaylıkla çözümlenebilir.

Haskell

- Kuvvetli tiplemeli, statik kapsam bağlamalı ve tip yorumlamalı bir dildir.
- Tam olarak fonksiyonel bir dildir. (değişkenler yoktur, atama ifadeleri yoktur, hiçbir çeşit yan etki yoktur).
- Tembel değerlendirme(lazy evaluation) kullanır (değer gerekmediği sürece hiçbir alt-ifadeyi değerlendirme)
- Liste kapsamları(list comprehensions), sonsuz listelerle çalışabilmeye izin verir

Örnekler

1.
```
fib 0 = 1
fib 1 = 1
fib (n + 2) = fib (n + 1)
            + fib n
```

2.
```
fact n
 | n == 0 = 1
 | n > 0 = n * fact (n - 1)
```

3. Liste işlemleri
   - Liste gösterimi:
   - `directions = ["north","south", "east", "west"]`
   - Uzunluk(Length): #

```
#directions = 4
```

   - .. operatorü ile aritmetik seriler

`[2, 4..10]gösterimi 2, 4, 6, 8, 10] olarak değerlendirilir.`

Örnekler devam

3. Liste işlemleri(devam)
   - ++ ile zincirleme(Catenation)

```
[10, 30] ++ [50, 70]→[10, 30, 50, 70]
```

   - : operatötü yoluyla

```
1:[3, 5, 7] → [1, 3, 5, 7]
```

```
product [] = 1
product (a:x) = a * product x

fact n = product [1..n]
```

Haskell örnekler (devam)

4. Liste kapsamları:küme gösterimi

```
[n * n | n ← [1..10]]
```

ilk 10 pozitif tamsayının karelerinden oluşan bir liste tanımlar

```
factors n = [i | i ← [1..n div 2],
                    n mod i == 0]
```

Bu fonksiyon verilen parametrenin bütün çarpanlarını hesaplar

Haskell örnekleri

- Quicksort(Hızlı Sıralama):

```
sort [] = []
sort (a:x) = sort [b | b ← x; b
  <= a]
 ++ [a] ++
sort [b | b ← x; b > a]
```

Haskell örnekleri (devam)

5. Tembel değerlendirme(Lazy evaluation)

```
positives = [0..]
squares = [n * n | n ← [0..]]
```

(sadece gerekli olanları hesapla)

```
member squares 16
```

True döndürür

Haskell örnekleri

- Üye(member) şu şekilde de yazılabilirdi:

```
member [] b = False
member(a:x) b=(a == b)||member x b
```

- Ancak, bu sadece kare(square) olan parametre tamkare olduğu zaman çalışacaktı; eğer değilse, sonsuza kadar üretmeye devam edecekti. Şu versiyon her zaman çalışır:

```
member2 (m:x) n
        | m < n= member2 x n
        | m == n    = True
        | otherwise = False
```

Fonksiyonel Dillerin uygulamaları

- LISP yapay zeka(artificial intelligence) için kullanılır
  - Bilgi gösterimi
  - Makine öğrenmesi(Machine learning)
  - Doğal Dil İşleme(Natural language processing), Konuşma ve görmeyi modelleme

Fonksiyonel –Emir Esaslı

Emir Esaslı (imperative) diller	fonksiyonel diller:
Verimli çalışma	Verimsiz çalışma
Karmaşık semantik(semantics)	Basit semantik(semantics)
Karmaşık sentaks(syntax)	Basit sentaks(syntax)
Eşzamanlılık(kullanıcı tanımlı)	Eşzamanlılık (otomatik)