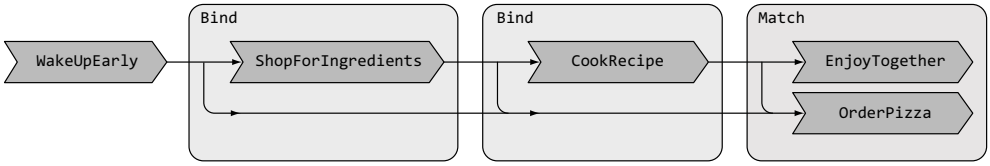
Pamiętamy z definicji `Bind`, że gdy stanem jest `Left`, wartość `Left` jest jedynie przekazywana. Zatem w poprzednim listingu, gdy mówimy `ComplainAbout(reason)`, `reason` jest tym, co nie udaje się w *którymkolwiek* z poprzednich kroków: jeśli nie uda nam się obudzić, `ComplainAbout` otrzyma tego powód. Podobnie będzie, gdy nie uda nam się zrobić zakupów i tak dalej.

Poprzedni diagram w kształcie drzewa jest poprawną schematyczną reprezentacją przepływu pracy. Inny sposób spojrzenia na to, bliższy szczegółom implementacji, jest pokazany na rysunku 6.2.



Rysunek 6.2 Łączenie w ciąg funkcji zwracających `Either`

Każda funkcja zwraca dwuczęściową strukturę `Either` i jest połączona z następną funkcją za pośrednictwem `Bind`. Przepływ pracy uzyskany przez łączenie kilku funkcji zwracających `Either` można uważać za system dwutorowy[3]:

- Istnieje tor *główny* (ścieżka powodzenia), idący z `R1` do `Rn`.
- Istnieje pomocniczy *tor równoległy* po stronie `Left`.
- Gdy jesteśmy na torze `Left`, pozostajemy na nim aż do końca drogi.
- Jeśli jesteśmy na torze `Right` z każdym zastosowaniem funkcji, albo kontynuujemy drogę po torze `Right`, albo zbaczamy na tor `Left`.
- `Match` jest na końcu toru, gdzie następuje rozłączenie równoległych torów.

Mimo że przykład z „ulubionym daniem" jest raczej błahy, jest reprezentatywny dla wielu scenariuszy programowania. Wyobraźmy sobie na przykład serwer bezstanowy, który odbierając żądanie, musi wykonać następujące kroki:

1. Zweryfikować żądanie.
2. Załadować model z bazy danych.
3. Dokonać zmian w modelu.
4. Zachować zmiany.

Każda z tych operacji może potencjalnie się nie powieść, a niepowodzenie w jednym kroku powinno uniemożliwić kontynuowanie przepływu pracy, odpowiedź powinna zaś zawierać szczegóły dotyczące sukcesu lub porażki żądanego działania.

Dalej przyjrzymy się użyciu `Either` w takim scenariuszu.

---

[3] W instrukcji do tego stylu obsługi błędów guru języka F#, Scott Wlaschin, tworzy analogię do „kolei". Zachęcam do przejrzenia jego artykułu „Railway Oriented Programming" i wideokonferencji dostępnej na stronie http://fsharpforfunandprofit.com/rop/.