Complément à deux : retenue et overflow (vue modulaire)

Idée générale

Avec n bits en complément à deux, on calcule les sommes *modulo* 2^n. Le résultat brut d'une addition est donc toujours dans $\{0, \ldots, 2^n - 1\}$. Puis on *interprète* ce code comme un entier signé dans

$$[-2^{n-1},\ 2^{n-1} - 1].$$

La "retenue" qui sort du bit de signe n'est que le débordement modulaire (le bit n-ième) : on la jette.

Quand la retenue peut être ignorée (pas d'overflow)

La retenue qui dépasse à gauche *n'est pas* un signe d'erreur. Elle est attendue en arithmétique modulaire. Il n'y a **pas** d'overflow si le résultat (interprété en signé) reste dans $[-2^{n-1}, 2^{n-1} - 1]$ et que le signe du résultat reste cohérent.

Exemple (4 bits) : retenue *avec* résultat correct

Sommons $7 + (-1)$:

$$\underbrace{0111}_{7} + \underbrace{1111}_{-1} = (1)\,0110.$$

On jette la retenue extérieure (1) (arithmétique mod 16) et on lit $0110 = 6$. En décimal, $7 + (-1) = 6$: tout est cohérent, **pas** d'overflow.

Lecture modulaire : $7 + 15 = 22 \equiv 6 \pmod{16}$, et $6 \in [-8, 7] \Rightarrow$ ok.

Autre exemple (4 bits) : deux négatifs, retenue mais pas d'overflow

$$\underbrace{1101}_{-3} + \underbrace{1110}_{-2} = (1)\,1011.$$

Lecture modulaire : $13 + 14 = 27 \equiv 11 \pmod{16}$. **Interprétation signée** : 1011 correspond à -5.

On obtient bien $-3 + (-2) = -5$. Il y a donc **pas d'overflow**, même si une retenue (1) apparaît.

Quand il y a overflow

Un *overflow* apparaît quand le résultat *modulo* 2^n sort de l'intervalle représentable signé : l'interprétation en complément à deux devient incohérente. Règle pratique :

Overflow ssi les deux opérandes ont le *même signe* et le signe du résultat est *différent*.

Exemple (4 bits) : deux positifs ⇒ résultat négatif

$$\underbrace{0100}_{4} + \underbrace{0101}_{5} = 1001.$$

En modulaire, $4 + 5 = 9 \equiv 1001 \pmod{16}$. Mais 1001 (signé, 4 bits) vaut -7. Deux positifs ⟶ résultat négatif ⇒ **overflow**.

Ici, il n'y a pas de retenue, montrant bien que *la retenue n'est pas le seul critère d'overflow.*

Exemple (4 bits) : deux négatifs trop grands (débordement côté −)

$$\underbrace{1100}_{-4} + \underbrace{1011}_{-5} = (1)\,0111.$$

Modulaire : $12 + 11 = 23 \equiv 7 \pmod{16}$. Interprétation signée : $0111 = +7$, alors qu'on additionne deux négatifs ⇒ **overflow**.

Pourquoi la retenue est "jetable" (vue modulaire)

L'addition en n bits calcule $a + b$ dans $\mathbb{Z}/2^n\mathbb{Z}$. Écrire $(1)\,r$ signifie $a + b = 2^n + r$ avec $0 \le r < 2^n$. Comme on travaille *modulo* 2^n, $2^n \equiv 0$ et seul r compte : on *ignore* la retenue. L'overflow ne vient pas du calcul (qui est correct mod 2^n) mais de l'*interprétation signée* lorsque $r \notin [-2^{n-1}, 2^{n-1} - 1]$.

Règles pratiques (à retenir)

- **La retenue qui sort du bit de signe est ignorée** (arithmétique modulaire).
- **Overflow** ⇔ mêmes signes en entrée, signe différent en sortie.

Résumé

En complément à deux, on additionne mod 2^n : la retenue extérieure est normale et jetée. L'overflow n'est pas une histoire de retenue, mais de *sortie de plage signée* : deux opérandes de même signe qui donnent un résultat de signe opposé.