The Boots Company synthesis of ibuprofen (1960's)

```
CC(C)Cc1ccccc1.CC(=O)OC(C)=O>[Al](Cl)(Cl)Cl>CC(=O)c1ccc(CC(C)C)cc1
CC(=O)c1ccc(CC(C)C)cc1.[Na+].CC[O-].CCOC(=O)CCl>>CCOC(=O)C1OC1(C)c2ccc(CC(C)C)cc2
CCOC(=O)C1OC1(C)c2ccc(CC(C)C)cc2.[OH3+]>>CC(C)Cc1ccc(C(C)C=O)cc1
CC(C)Cc1ccc(C(C)C=O)cc1.NO>>CC(C)Cc1ccc(C(C)C=NO)cc1
CC(C)Cc1ccc(C(C)C=NO)cc1>>CC(C)Cc1ccc(C(C)C#N)cc1
CC(C)Cc1ccc(C(C)C#N)cc1.{2}O>>CC(C)Cc1ccc(C(C)C(=O)O)cc1
```

```python
from rxnSMILES4AtomEco import atom_economy

#use triple quotes (""") to define a multiline string
reactions_smiles = """CC(C)Cc1ccccc1.CC(=O)OC(C)=O>Al(Cl)Cl>CC(=O)c1ccc(CC(C)C)cc1
CC(=O)c1ccc(CC(C)C)cc1.[Na+].CC[O-].CCOC(=O)CCl>>CCOC(=O)C1OC1(C)c2ccc(CC(C)C)cc2
CCOC(=O)C1OC1(C)c2ccc(CC(C)C)cc2.[OH3+]>>CC(C)Cc1ccc(C(C)C=O)cc1
CC(C)Cc1ccc(C(C)C=O)cc1.NO>>CC(C)Cc1ccc(C(C)C=NO)cc1
CC(C)Cc1ccc(C(C)C=NO)cc1>>CC(C)Cc1ccc(C(C)C#N)cc1
CC(C)Cc1ccc(C(C)C#N)cc1.{2}O>>CC(C)Cc1ccc(C(C)C(=O)O)cc1"""
atom_economy(reactions_smiles)
```

```python
from rxnSMILES4AtomEco import get_atom_economy
# value = get_atom_economy(reactions_smiles)
# print(value)

# Split reactions_smiles into lines
lines = reactions_smiles.splitlines()

# Create a dictionary to store the results
results_individual = {}

# Loop through each line and call get_atom_economy for each one
for i, line in enumerate(lines):
    step_key = f"Step {i+1}"  # Create Step 1, Step 2, ..., Step n
    results_individual[step_key] = get_atom_economy(line)  # Call your function and store the

# Now, 'results' contains the results for each line, e.g., results["Step 1"], results["Step 

# Example: Print all results
for step, result_individual in results_individual.items():
    print(f"{step}: {result_individual:.1f}%")
```

```python
# Split reactions_smiles into lines
lines = reactions_smiles.splitlines()

# Create a dictionary to store the results
results_cumulative = {}

# Loop through each incremental combination of lines and call get_atom_economy
for i in range(1, len(lines) + 1):  # Start from 1 to n
    # Get the first i lines
    combined_reactions = "\n".join(lines[:i])

    # Call get_atom_economy with the combined lines up to the current step
    step_key = f"Step_{i}"
    results_cumulative[step_key] = get_atom_economy(combined_reactions)  # Store the result

    # Example: Print the current step's result
    print(f"{step_key}: {results_cumulative[step_key]:.1f}%")
```

```python
import matplotlib.pyplot as plt

# Extract the steps (x-axis) and Atom Economy values (y-axis) for both individual and cumula
steps = list(results_individual.keys())  # Step names: ['step_1', 'step_2', ...]
individual_values = list(results_individual.values())  # Atom Economy values for individual 
cumulative_values = list(results_cumulative.values())  # Atom Economy values for cumulative 

# Calculate the Byproduct(s) as the difference between 100% and the desired product
byproduct_values = [100 - value for value in individual_values]

# Create a single plot
fig, ax = plt.subplots(figsize=(10, 7))  # Single axis for both plots
plt.rcParams.update({'font.size': 16})

# Plot the stem plot for cumulative Atom Economy values in black
```

```
ax.stem(steps, cumulative_values, linefmt='k:', markerfmt='k>:', basefmt=" ", label="Cumulati

# Bar plot for individual Atom Economy values
bars = ax.bar(steps, individual_values, color='#009E73', label="Desired product", alpha=0.8)
ax.bar(steps, byproduct_values, bottom=individual_values, color='#E69F00', label="Byproduct(s

# Add the text (value of desired product) in white in the middle of each bar
for i, bar in enumerate(bars):
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width() / 2, height / 2, f'{height:.1f}%',
             ha='center', va='center', color='white', fontweight='bold')

# Add labels and titles
ax.set_title('Boots Atom Economy: Individual- and Cumulative Steps\n\n\n')
ax.set_xlabel('\nSteps')
ax.set_ylabel('Atom Economy (%)')
ax.set_ylim(0, 100)  # Set Y-axis from 0 to 100%
#ax.tick_params(axis='x', rotation=45)  # Rotate x-axis labels for better readability

# Add the legend
ax.legend(loc='upper center', bbox_to_anchor=(0.5, 1.2), ncol=3,reverse=True)

# Improve layout to avoid label overlap
plt.tight_layout()

# Show the plot
plt.show()

# Save the plot
#save_path = '/tmp/Boots_histogram.png'  # Writable on many systems
#plt.draw()  # Force render
#print(f"Saving to: {save_path}")
#plt.savefig(save_path, dpi=300, bbox_inches='tight')
#plt.close()
#print("Save completed (check directory).")
```